iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.83214

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

Cloud Honeypot Monitoring System: A Framework for Real-Time Attack Detection and Behavioral Analysis in Cloud Environments

Krishnanshu Khilare, Sakshi Dange, Apeksha Gotmare, Mansi Shinde
Department of Computer Engineering Sinhgad College of Engineering, Savitribai Phule Pune University Pune, Maharashtra, India

Abstract—The accelerating adoption of cloud computing has fundamentallyexpandedthedigitalattacksurface,makingcloud infrastructuresanincreasinglyattractivetargetforsophisticated adversaries. Conventional security mechanisms such as firewalls, intrusion detection systems, and antivirus platforms are primarily reactive in nature, offering limited visibility into attacker methodologies, behavioral progression, and exploitation tactics once initial contact occurs. To bridge this critical knowledge gap, this paper presents the design, implementation, and evaluation of a Cloud Honeypot Monitoring System deployed on Amazon Web Services that attracts, records, and analyzes malicious interactions within a fully isolated and controlled cloud environment. The system deploys decoy services emulating SSH, HTTP, and HTTPS protocols on AWS EC2 instances residing within an isolated VirtualPrivateCloud,andintegratesAWSCloudWatchandS3 forreal-timelogcaptureandpersistentstorage.APython-based backend constructed with Flask provides log parsing, event correlation, signature matching, and rule-based anomaly detection capabilities. A React and D3.js frontend dashboard renders live attack feeds, geographic distribution maps, attack type distribution charts, and complete attack lifecycle visualizations.Controlledattacksimulationsareexecutedusing Nmap, Metasploit, and custom Python scripts to generate realistic threat scenarios encompassing port scanning, brute- force credential attacks, reconnaissance activities, and exploitationattempts.Thesystemsuccessfullydemonstratedthe ability to attract, capture, process, and visualize attacker interactions in real time, while supporting exportable threat intelligence reports in JSON and CSV formats.

Keywords—cloud security, honeypot, AWS EC2, attack simulation, anomaly detection, intrusion analysis, threat intelligence,logcorrelation,CloudWatch,brute-forcedetection.

I. INTRODUCTION

A. Background

Cloud computing has emerged as the dominant paradigm formodernenterpriseinfrastructure,enablingscalable,flexible, and cost-efficient deployment of applications across geographically distributed environments. Organizations across sectors increasingly entrust cloud platforms with the storage of sensitive data, execution of critical business logic, and delivery of customer-facing services. This rapid and pervasive adoption has simultaneously expanded the attack surface available to adversarial actors. Attackers persistently probe cloud environments, targeting misconfigurations in network access control,authenticationweaknessesinAPIgateways, vulnerabilities within virtualized compute layers, and unprotected storage endpoints. Unlike traditional on-premise architectures, cloud platforms operate under shared responsibilitymodels,introducingnewcategoriesofownership ambiguity and misconfiguration risk that adversaries are well- equipped to exploit.

Thechallengeofunderstandingadversarialbehaviorwithin cloud environments is one of the most pressing open problems in applied cybersecurity research. Security instrumentation that focuses exclusively on blocking or filtering known malicious patterns provides little insight into how attackers discover targets, execute reconnaissance, escalate privilege, or persist within compromised systems. Furthermore, modern attack campaigns leverage automated botnets, distributed scanning infrastructure, and adaptive exploitation toolchains that evolve faster than signature databases can be updated.

B. TheRoleofHoneypotsinSecurityResearch

Honeypot technology provides a fundamentally different approach by shifting from detection-and-blocking to observation-and-understanding.

A honeypot is a deliberately exposed decoy system designed to attract unauthorized interaction and record every aspect of the resulting exchange. Because honeypots have no legitimate production traffic, any interaction they receive is inherently suspicious, enabling high- confidence threat observation with minimal false positives. When deployed in cloud environments, honeypots can emulate real services including SSH daemons, HTTP web servers, and HTTPS endpoints, presenting convincing targets to attackers while keeping actual production infrastructure entirely safe.

Cloud-based honeypot deployments offer particular advantages over traditional on-premise approaches. Cloud infrastructure enables rapid provisioning of isolated virtual machines, fine-grained network access control through Virtual Private Clouds and security groups, native log aggregation through services such as CloudWatch, and scalable storage for captured event data through services such as S3.

C. ResearchMotivationandGap

Despite the availability of honeypot frameworks in academic literature and open-source tooling, integrated platforms combining cloud-native deployment, real-time log processing, multi-protocol honeypot emulation, structured detection algorithms, and interactive visualization remain underrepresented,particularlyinresearchtargetingAWS-based cloudenvironments.Existingworkfrequentlyaddressesisolated aspects without providing a unified, deployable, end-to-end observable system.

D. ObjectivesandContributions

This paper addresses the identified gap by presenting a complete Cloud Honeypot Monitoring System with the following objectives: (i) deploy isolated multi-protocol honeypotsonAWSEC2withinasecuredVirtualPrivateCloud;

(ii) capture all attacker interactions through CloudWatch and persist them in S3; (iii) implement a Python-based backend pipeline for log parsing, event correlation, signature matching, and anomaly detection; (iv) develop an interactive React dashboard for real-time visualization; and (v) validate the system through controlled attack simulations using Nmap, Metasploit, andcustomscripts.Theprimarycontributionsareadeployable, modular cloud honeypot architecture with formal mathematical underpinning; an integrated detection engine; a complete data pipeline from raw CloudWatch events to structured threat intelligence; and a validated experimental platform for repeatable cloud security research.

II. LITERATURE REVIEW

A. AttackDetectioninCloudVirtualEnvironmentsUsing Honeypots

PanditreandGaikwad[1]addressedrisingsecurityrisksin cloud virtual machine environments caused by distributed denial-of-service attacks, vulnerability scanning, port exploitation, and coordinated multi-stage intrusion campaigns. Theauthorsobservedthattraditionalintrusiondetectionmodels perform poorly in cloud deployments due to virtual machine mobility and encrypted network traffic. They proposed the NICE (Network Intrusion detection and Countermeasure sElection) framework,whichcombinesnetworkattackgraphconstruction, virtual machine profiling, real-time traffic analysis, and automated honeypot redirection. The framework monitors suspicious behavior, estimates risk probability, and transparently redirects attackers to honeypot instances for deeper behavioral observation.

The NICE framework demonstrates strong conceptual alignment between attack graph modeling and honeypot redirection. However, it relies on predefined attack graph templates, limiting its ability to detect previously unseen intrusion paths. It does not provide interactive dashboards or exportable threat intelligence artifacts, and its evaluation is confined to simulated environments rather than live cloud deployments on commodity infrastructure.

B. HoneypotMethodtoLureAttackersWithoutHolding Crypto-Assets

Uchibori et al. [2] presented a deception architecture specifically designed to attract financially motivated attackers targetingblockchainandcryptocurrencyinfrastructure.Thecore innovation involves returning legitimate high-balance third- party wallet addresses in RPC responses to make the honeypot appear highly valuable while eliminating actual financial risk. The architecture incorporates RPC proxy nodes, multi-node synchronization, cryptographic signature logging, and behavioral clustering of attacker activity based on method execution patterns. This work makes a notable contribution in applying deception technology to a previously underexplored attackdomain.However,thesystemisnarrowlyspecializedfor cryptocurrency infrastructure and does not generalize to broad cloud service protocols such as SSH or HTTP.

C. ASurveyofHoneypotsandHoneynetsforIoT,IIoT,and Cyber-Physical Systems

Franco et al. [3] conducted a comprehensive survey spanning two decades of honeypot and honeynet research applied to Internet of Things, Industrial Internet of Things, and cyber-physical system domains. The survey systematically categorizes honeypots by interaction level and examines representative systems including IoTPOT, FIRMADYNE, HoneyThing, and Conpot in terms of device emulation fidelity, detected attack types, and analytical capability. Key challenges identified include resistance to fingerprinting, realistic device emulation, adaptive reconfiguration, and large-scale deployment. The survey provides an invaluable taxonomy but does not address cloud-specific deployments or examine integration with cloud-native logging and storage services.

D. ComparativeAnalysis

Across these three works, a consistent pattern emerges: honeypotresearchhasproducedeffectivetechniquesforspecific domainsbuthasnotconvergedonaunified,cloud-native,multi- protocol platform combining real-time monitoring, structured detection, and interactive threat intelligence visualization. The NICE framework [1] operates closest to the target domain but lacks visualization and live cloud instrumentation. The cryptocurrency honeypot [2] demonstrates sophisticated behavioral profilingbutistoodomainspecific.TheIoTsurvey[3]identifiesdesignchallengesapplicabletoclouddeploymentsbutdoesnotaddressthemdirectly.Theproposedsystem addresses these gaps by delivering a fully integrated, AWS- native, multi-protocol honeypot monitoring platform with end- to-end observability.

III. PROBLEM STATEMENT

Cloud computing environments are under continuous and escalatingthreatfromsophisticated,well-resourcedadversaries. Despite significant investment in perimeter security tooling, organizations relying on conventional defenses remain largely blind to the behavioral dimension of adversarial activity. Firewalls enforce access control but provide no intelligence about the techniques employed by actors who probe their boundaries. Intrusion detection systems raise alarms based on known signatures but fail silently against novel attack patterns or evasion techniques.

The fundamental problem is the absence of a secure, controlled, and instrumented environment in which malicious activities targeting cloud services can be safely attracted, observed in full detail, and analyzed systematically without placing production systems at risk. Without such an environment, security researchers cannot empirically characterize how attackers discover cloud endpoints, what credential combinations they attempt, how they sequence reconnaissanceandexploitationactivities,orwhatpayloadsthey deliver to newly compromised systems.

Existingapproachesexhibitthreeprimarylimitations.First, they are predominantly reactive, generating alerts only after maliciousactivityhasbegun.Second,theyoperateinproduction environments where real user traffic constrains the depth of instrumentation. Third, they do not support controlled experimentation, making it impossible to validate detection mechanismsagainstreproduciblethreatscenarios.Theproposed Cloud Honeypot Monitoring System directly addresses these limitations by providing a dedicated, isolated, and fully instrumented research environment.

IV. PROPOSED SYSTEM

A. SystemOverviewandWorkflow

The Cloud Honeypot Monitoring System operates as a structured pipeline comprising six functionally distinct stages that together transform raw attacker interactions into structured threat intelligence. The pipeline begins when external actors interact with exposed honeypot services. These interactions are logged at the service level, captured at the network level, forwarded to cloud-native aggregation services, processed by a backend analytical engine, persisted in structured storage, and finally rendered through an interactive visualization dashboard.

Honeypot instances running on AWS EC2 within an isolated VPC expose SSH, HTTP, and HTTPS services with intentionally weak configurations designed to attract unauthorized access attempts. Every connection attempt, authentication trial, HTTP request, and payload delivery is recorded locally and simultaneously forwarded to AWS CloudWatchlogstreams.CloudWatchaggregateseventsinreal time and periodically exports complete log archives to designated S3 buckets for long-term persistence. A Python- based backend service running Flask continuously ingests logs, applies parsing and normalization routines, and passes records throughadetectionengineperformingsignaturematching,event correlation, and anomaly scoring.

B. CloudInfrastructure

The system's cloud infrastructure is anchored by an AWS Virtual Private Cloud providing complete network isolation. Within this VPC, dedicated subnets host the honeypot EC2 instances, restricting all inter-subnet communication to explicitly permitted flows. Internet-facing exposure is deliberatelycontrolled:honeypotinstancesadvertiseserviceson standardportstoattractinternet scannersandtargetedattackers, while all management and backend traffic is restricted to authorized administrative channels. AWS IAM roles and policies govern access to CloudWatch log streams, S3 buckets, and other AWS resources.

C. HoneypotDeploymentandTrafficCapture

SSH honeypots expose port 22 with fake credential acceptance logic that logs all attempted username-password combinations without granting genuine access. HTTP and HTTPS honeypots expose common web application endpoints including login pages, administrative interfaces, and API paths frequently targeted by web application scanners. Service banners advertise software versions commonly associated with exploitable vulnerabilities. Monitoring agents capture system- level events and transmit them to CloudWatch in real time. Network-level capture using VPC Flow Logs provides a complementary record enabling cross-validation between application-layer and network-layer event records.

D. BackendProcessingandDetectionEngine

The backend is implemented in Python using Flask and exposesacomprehensiveRESTAPIforlogingestion,analytics retrieval, and report generation. The detection engine applies a multi-stageanalysisprocess:signaturematchingcomparesevent fieldsagainstk nownattackpatterns;acorrelationenginegroups eventsbysourceIPandtemporalproximitytoreconstructmulti- stepattack sequence s;andananomaly scoringmodulecomputes weighted composite scores and triggers alerts when scores exceed configurable thresholds.

E. DashboardandThreatIntelligenceGeneration

The React-based frontend dashboard provides real-time visibility into all aspects of honeypot activity. The dashboard displays a global statistics panel showing total attacks, unique attacker IP addresses, and event counts over the past hour and twenty-fourhours.Anattacktypedistributionchartbreaksdown detectedincidentsbycategory.Aliveattackfeedpaneldisplays recenteventsinrealtimewithsourceIPaddressesandattempted credentials. Threat intelligence export functionality serializes recorded attack events into JSON or CSV format, enabling integration with external platforms and offline analysis.

V. SYSTEM ARCHITECTURE

Thesystemarchitecturefollowsalayered,modulardesign in which each functional layer communicates with adjacent layers through well-defined interfaces. The architecture is organized into the following distinct layers:

ExternalInteractionLayer:Internet-accessibleorsimulated attack sources interact with honeypot services exposed by EC2 instances at the network perimeter of the AWS VPC. Security group rules define which ports are reachable from external sources, controlling the attack surface presented to potential adversaries.

Honeypot and Service Emulation Layer: EC2 instances runningUbuntuServerhostSSH,HTTP,andHTTPShoneypot services. Each service emulator is configured with fake credentials, intentionally exploitable endpoint configurations, and comprehensive interaction logging.

Traffic Capture and Ingestion Layer: The CloudWatch agent deployed on each honeypot instance forwards structured log events to dedicated CloudWatch log groups in real time. VPC Flow Logs provide supplementary network-level records. S3 receives periodic log archive exports, providing durable long- termstorage. An Attack Orchestrator component within the VPC hosts the Nmap and Metasploit toolchain for controlled attack simulation.

Backend Processing Layer: The Flask-based backend ingestionserviceretrieveseventsfromCloudWatchandS3 through the AWS SDK. A log parsing pipeline normalizes raw events. The Signature Engine performs pattern matching. The Correlation Engine groups related events into sessions. The Anomaly Scorer computes composite threat scores and generates alert records when thresholds are exceeded.

StorageandIndexingLayer:Rawlogblobsarepersistedin S3. Structured, processed event records and alert objects are stored in a local database accessible to the backend API. A background thread periodically uploads newly processed records to S3 for redundant archival.

Visualization and Export Layer: The API Layer exposes REST endpoints for attack records, statistics, geographic distribution data, alerts, and export operations.

The React dashboard renders results using D3.js and Chart.js. Export modules serialize records to JSON or CSV on demand.
Thecompletedataflowproceedsas:attackerinteraction→honeypotservice→CloudWatchlogstream→S3rawstorage→backendingestion →parsingandnormalization→signature matching → event correlation → anomaly scoring → alert generation → structured storage → REST API → React dashboard rendering.

Fig.1.SystemArchitectureoftheCloudHoneypotMonitoring Systemshowingthelayereddataflowfromexternal attackers through AWS infrastructure to the visualization dashboard.

VI. MATHEMATICALMODEL

A. SetsandBasicDefinitions

Let T denote the discrete timeline of system operation, whereeachelementt ∈Trepresentsadiscretetimestamp.LetH={h_1,h_2,…,h_n}bethesetofdeployedhoneypotinstances,andletSrepresentthesetofservicetypes,whereS⊇ {SSH,HTTP, HTTPS}.LetAdenotethesetofattackerIPaddresses,Rtheset of raw events generated by honeypot interactions, L the set of structured log entries produced by the parsing pipeline, and D theset ofdetectedincident records(alerts). Let Pdenotetheset ofdetectionpatternscomprisingknownattacksignatures,andM the set of detection model instances including both rule-based detectors and statistical anomaly scorers.

B. EventExtractionFunction

The event extraction function E is defined as: E : R × H × T → L. For a raw event r ∈ R observed at timestamp t on honeypotinstanceh∈ H,thefunctionE(r,h,t)=l∈ Lproduces a structured log entry containing: {id, timestamp, honeypot_id, service_type, src_ip, dest_port, payload_hash, event_type, raw_blob_reference}. This function represents the normalizationstagetransformingunstructuredservice-levellog data into records amenable to pattern matching and statistical analysis.

C. HoneypotStateModel

Each honeypot instance h ∈ H maintains a state s_h(t) drawn from the state space Σ = {Idle, Listening, Interacted, Compromised_simulated}. State transitions are governed by: δ_h : Σ × L → Σ. The transition δ_h(Listening, l) = Interacted occurs when l.event_type ∈ {connection_attempt, authentication_request, http_request}. The transition δ_h(Interacted, l) = Compromised_simulated is triggered whenl contains indicators of successful simulated exploit patterns, such as post-authentication command execution logs.

D. LogAggregationandCorrelation

ThecorrelationfunctionC:$\wp$(L)→$\wp$(L)partitionsthelog set L into correlated groups by clustering entries sharing the same src_ip, session identifier, or temporal window. The temporalwindowattimetis:W_t={l∈L|t−Δ≤l.timestamp ≤ t}, where Δ is the configurable correlation window duration (nominallyfiveminutes).Thecorrelationfunctiongroups entrieswithinW_tbysourceattributes,producingsession-level eventclustersrepresentingcompleteinteractionsequencesfrom a single attacker source.

E. DetectionPredicates

For each detection pattern p ∈ P, a binary predicate is defined as: predicate_p : $\wp$(L) → {0, 1}. For a correlated event group G ⊆ L, predicate_p(G) = 1 indicates the group satisfies conditions associated with pattern p. For statistical anomaly detection models m ∈ M, a continuous anomaly score function score_m : $\wp$(L) → ℝ is defined, where detection occurs when score_m(G) ≥ θ_m.

F. AlertGenerationandOptimization

Thealertgenerationfunctionis:A_G:$\wp$(L)×P×M→D. For a correlated group G, the function A_G(G, p, m) = d produces an alert record d containing: {alert_type, evidence, severity,timestamp}.Thedesign-timeoptimizationobjectiveis: maximize Utility(θ) = w_1 · Coverage(θ) − w_2 · FalsePositiveRate(θ)−w_3·Cost(θ),whereCoverage(θ)=|{true incidents detected}| / |{true incidents}| and Cost(θ) captures computational overhead, storage consumption, and human review effort.

VII. IMPLEMENTATION

A. AWSVPCandNetworkConfiguration

The implementation begins with provisioning an AWS Virtual Private Cloud configured to provide complete network isolation. The VPC is assigned a dedicated CIDR block with separate subnets for honeypot instances, backend processing nodes, and management traffic. Security groups are defined for each instance type: honeypot security groups permit inbound TCP traffic on ports 22, 80, and 443 from any source to attract unsolicited interaction, while restricting all other inbound traffic. Backend security groups permit inbound communication only from the honeypot subnet and management CIDR. AWS IAM roles are assigned to each EC2 instance using least-privilege policies.

B. EC2InstanceDeploymentandHoneypotConfiguration

Multiple EC2 instances running Ubuntu Server are deployed within the isolated honeypot subnet. Each instance is provisioned with the CloudWatch agent configured to forward application and system logs to dedicated log groups. The SSH honeypot service listens on port 22, accepts all connection attempts, and records all usernames and passwords submitted during authentication without granting genuine access. HTTP and HTTPS services expose a simulated web application with endpointscommonlytargetedbyautomatedscanners,including loginpages,administrativepanelURLs,configurationfilepaths, and API endpoints with intentionally weak configurations.

C. LoggingInfrastructure

AWS CloudWatch is configured with one log group per honeypot service type. CloudWatch subscription filters automatically export log events to designated S3 buckets at regular intervals, with each export object prefixed by date and honeypot identifier for organized retrieval. S3 bucket policies enforce server-side encryption and restrict access to the designated backend IAM role, ensuring raw log data is durably preserved independent of the real-time processing pipeline.

D. BackendAPIandProcessingPipeline

The backend is implemented using Python and the Flask microframework. On startup, the application initializes a local SQLite database containing tables for attack records, session records, alert records, and report metadata. The log ingestion pipeline retrieves log entries from CloudWatch and S3, normalizes raw events into the standard schema, and persists structuredrecordstothelocaldatabase.Thedetectionpipeline applies the signature engine, correlation engine, and anomaly scorer in sequence. A background thread periodically uploads newly processed records back to S3 for redundant archival.

TheRESTAPIexposesendpointsincluding:GET /api/attacks for retrieving all attacks with optional filtering by attack type; GET /api/attacks/recent for recent attacks within a configurable hour window; GET /api/analytics/stats for aggregatestatisticsincludingtimelinedata;GET /api/analytics/geo for geographic distribution by country and city; GET /api/alerts for recent alert listings; POST /api/export fordataserializationinJSONorCSV;POST/api/reports/generateforstructuredsecurityreportgeneration; and POST /api/aws/s3/upload for manual S3 upload triggering.

E. FrontendDashboard

The frontend is implemented as a single-page React application communicating with the Flask backend through HTTP polling. The statistics panel displays aggregate counters fortotalattacks,uniqueattackerIPaddresses,andlast-hourand last-twenty-four-houreventcounts.Theattacktypedistribution panelrendersapiechartusingD3.jsbreakingdownincidentsby categoryincludingSSHbruteforce,credentialharvesting,SQL injection, path traversal, and web application attacks. The live attack feed panel continuously updates with recent events showingattacktypelabels,sourceIPs,andattemptedcredential pairs. An attack timeline chart displays hourly event frequency enablingidentificationofcampaignburstsandperiodicscanning cycles.

Fig.2.CloudHoneypotMonitoringDashboardshowingtotal attackstatistics,attacktypedistributionchart,andliveattack feed panel.

VIII. EXPERIMENTALSETUP

A. AttackSimulationEnvironment

ValidationoftheCloudHoneypotMonitoringSystemwas conducted through controlled attack simulations executed entirely within the isolated AWSVPCenvironment. The attack orchestrator EC2 instance served as the source of all simulated malicioustraffic,directingactivitiesexclusivelyatthedeployed honeypot instances.

This arrangement ensured all simulated attacks were captured by the honeypot logging infrastructure without any risk of impact on external systems.

B. Nmap-BasedNetworkReconnaissance

Networkreconnaissancesimulationswereconductedusing Nmaptoreplicateinitialscanningbehaviorcommonlyexhibited by automated internet scanners and targeted attackers. Nmap scans were executed against honeypot IP addresses to perform TCP SYN port discovery, service version detection, and operating system fingerprinting. These scans generate characteristic patterns in network logs: rapid sequential connection attempts across multiple ports from a single source IP within a short time window, detectable by the correlation engine as reconnaissance behavior.

C. Metasploit-BasedExploitationSimulation

Metasploit Framework was used to simulate targeted exploitation attempts against exposed honeypot services. SSH brute-force modules were executed against the SSH honeypot, generating high volumes of authentication attempts using common credential wordlists including default username- password combinations and dictionary-derived entries. HTTP exploitation modules probed the web application honeypot for SQL injection, path traversal, and directory enumeration vulnerabilities, each producing distinct log patterns detectable by the signature engine.

D. CustomPythonAttackScriptsandValidation

CustomPythonscriptsweredevelopedtosimulatespecific attack scenarios providing greater control over parameters and timing.Thesescriptssimulatedcredentialstuffingattacksusing email-format usernames, multi-stage HTTP attacks sequencing reconnaissance and authentication bypass, and slow-rate brute- forceattacksdesignedtotesttemporalwindowparametersofthe correlation engine. The system was validated through unit testing of individual components, integration testing of end-to- end data flow, acceptance testing against functional requirements,andcloud-specifictestingofVPCisolation,IAM role restrictions, and CloudWatch subscription filter operation.

IX. RESULTSANDDISCUSSION

A. SystemDeploymentandOperationalValidation

TheCloudHoneypotMonitoringSystemwassuccessfully deployed and validated within an isolated AWS environment. All honeypot services were confirmed operational and accessible from the attack orchestrator instance. CloudWatch loggroupsreceivedcontinuouseventstreamsfromallhoneypot instances, and S3 bucket exports confirmed that log data was being durably persisted. The Flask backend API responded correctly to all configured endpoints, and the React dashboard successfully populated with data retrieved from the API, confirming end-to-end pipeline integrity.

B. AttackCaptureandLogProcessing

During testing, controlled simulations generated a diverse set of attacker interactions that were captured, processed, and storedbythesystem.TheSSHbrute-forcesimulationsproduced large volumes of failed authentication events, each containing source IP, attempted username, attempted password, and timestamp fields, all correctly parsed and normalized by the backend ingestion pipeline. The dashboard aggregate statistics panel indicated a cumulative total of 45 recorded attack events originating from a single unique source IP address in the local simulation environment.

Attack type distribution analysis revealed that SSH brute forceaccountedforthedominantproportionofobservedevents at40%,followedbycredentialharvestingat20%,SQLinjection probing at 16%, path traversal at 9%, and a distribution of configuration file access, administrative access, cross-site scripting, and WordPress-specific probing attempts comprising theremainder.Theseproportionsareconsistentwiththerelative frequencies used in simulation scripts and validate the correctness of the signature matching and event classification logic.

C. DetectionEnginePerformance

The detection engine successfully identified all major attackcategoriespresentinthesimulatedtraffic.Repeatedfailed authenticationsequenceswerecorrectlyclassifiedasbrute-force behavior through the credential attempt frequency signature pattern.HTTPrequestsequencestargetingsequentialURLpaths were correctly identified as path traversal and directory enumeration activity. SQL injection attempts were detected through payload string matching.

The correlation engine successfully grouped related events from the same source IP withinconfiguredtemporalwindows,enablingreconstructionof complete attack sequences spanning reconnaissance, credential guessing, and web application probing phases.

D. DashboardAnalyticsandVisualization

The dashboard visualization layer performed correctly across all tested scenarios. The attack type distribution chart rendered an accurate proportional breakdown of all event categories. The live attack feed confirmed near-real-time visibility into attack events, with new events appearing within the polling interval following their ingestion. Performance testingindicatedthatthesystemhandledcontinuouslog ingestion from simulation-generated traffic without observable processing delays. The Flask API maintained stable response times across all endpoint types.

Performance testing indicated that the backend API maintained stable response times across all endpoint types. The modular architecture proved scalable, allowing additional honeypot instances and services to be integrated without major architectural changes. [Experimental measurements for precise throughputfigures,latencydistributions,anddetectionaccuracy metrics such as false positive rate and true positive rate are unavailable from the source report. No labeled ground truth dataset with known attack/benign record counts was presented from which precision, recall, or F1 scores could be derived.]

Fig.3.LiveAttackFeedandTopAttackersPanelshowingSSH brute-force events with source IP addresses and attempted credentials.

X. ADVANTAGES

The Cloud Honeypot Monitoring System offers several substantive advantages over conventional security monitoring approaches. By deploying decoy services in an isolated cloud environment, the system enables deep observation of attacker behaviorwithoutexposinganyproductioninfrastructuretorisk, providing a safe research substrate impossible to replicate in a live production environment. The integration of AWS CloudWatchandS3providesenterprise-gradelogdurabilityand scalability, ensuring that no interaction is lost even under high- volume attack conditions.

Themodulararchitectureenablesindividualcomponentsto be upgraded or replaced without disrupting the overall system. The multi-protocol honeypot coverage spanning SSH, HTTP, and HTTPS allows the system to attract and characterize a diverse population of adversarial actors. The combination of signature-based and statistical anomaly detection provides broader coverage than either approach alone. The exportable threatintelligencecapabilityenablesgatheredintelligencetobe shared with external platforms and research communities, multiplying the defensive value beyond the immediate deployment context.

XI. LIMITATIONS

Thecurrentimplementationcarriesseverallimitationsthat bound its applicability and analytical depth. The detection engine relies exclusively on rule-based signature matching and threshold-based anomaly scoring, which are constrained by the completeness of the signature library and accuracy of manually configured thresholds. Novel attack techniques not matching existingsignaturesandsubtleanomaliesbelowthresholdvalues will not be detected.

The system operates as a single-cloud deployment limited to AWS, precluding comparative analysis of attack patterns across different cloud provider environments. The absence of automated alerting mechanisms means analysts must actively monitorthedashboardtobecomeawareofhigh-severityevents. The system provides no automated response or mitigation capability;itsscopeisstrictlyobservationalandanalytical.The experimental validation conducted with controlled simulations prevents empirical characterization of false positive rate, detection latency, and throughput limits under realistic attack volumes.

XII. FUTUREWORK

The Cloud Honeypot Monitoring System establishes a robust foundation extensible in several high-impact directions. The most significant near-term enhancement involves integratingmachinelearning-baseddetectiontocomplementthe existingrule-basedengine.Statisticalmodelsincludingisolation forests,autoencoders,andclusteringtechniquescanlearn complex behavioral patterns from historical attack logs and identify deviations from normal attacker behavior that elude rule-basedsystems.Supervisedclassificationmodelstrainedon labeledattackdatasetscoulddramaticallyreducefalsepositives, while deep learning sequence models could capture temporal attack progression patterns invisible to point-in-time signature matching.

Automated alerting integration through email, SMS, or collaboration platforms such as Slack would eliminate the requirement for continuous dashboard monitoring, enabling prompt human response to high-severity events.

Threat intelligence feed integration connecting the detection engine to IP reputation databases, malware signature feeds, and open-source intelligence sources would enable contextual annotation of attacker IP addresses with known associations to botnet infrastructure and threat actor groups. Expanding the diversity of emulated services would broaden the population of adversarial actors attracted. Adding honeypots emulating cloud storage APIs, container orchestration interfaces, IoT device protocols, relational database services, and serverless function endpoints would enable observation of attack techniques entirely absent from SSH and HTTP-focused deployments. Deploying distributed honeypot instances across multiple cloud providers including Google Cloud Platform and Microsoft Azure would enable comparative studies of attack pattern distributions across different cloud ecosystems.

Infrastructure-as-Code automation using Terraform or AWS CloudFormation would transform the current manual deployment into a fully automated, reproducible provisioning pipeline. Enhanced visualization features including MITRE ATT&CKframeworkmappingofobservedattackertechniques, predictive analytics for attack likelihood forecasting, and interactive attack chain graph exploration would significantly improve analytical utility for experienced security researchers.

XIII. CONCLUSION

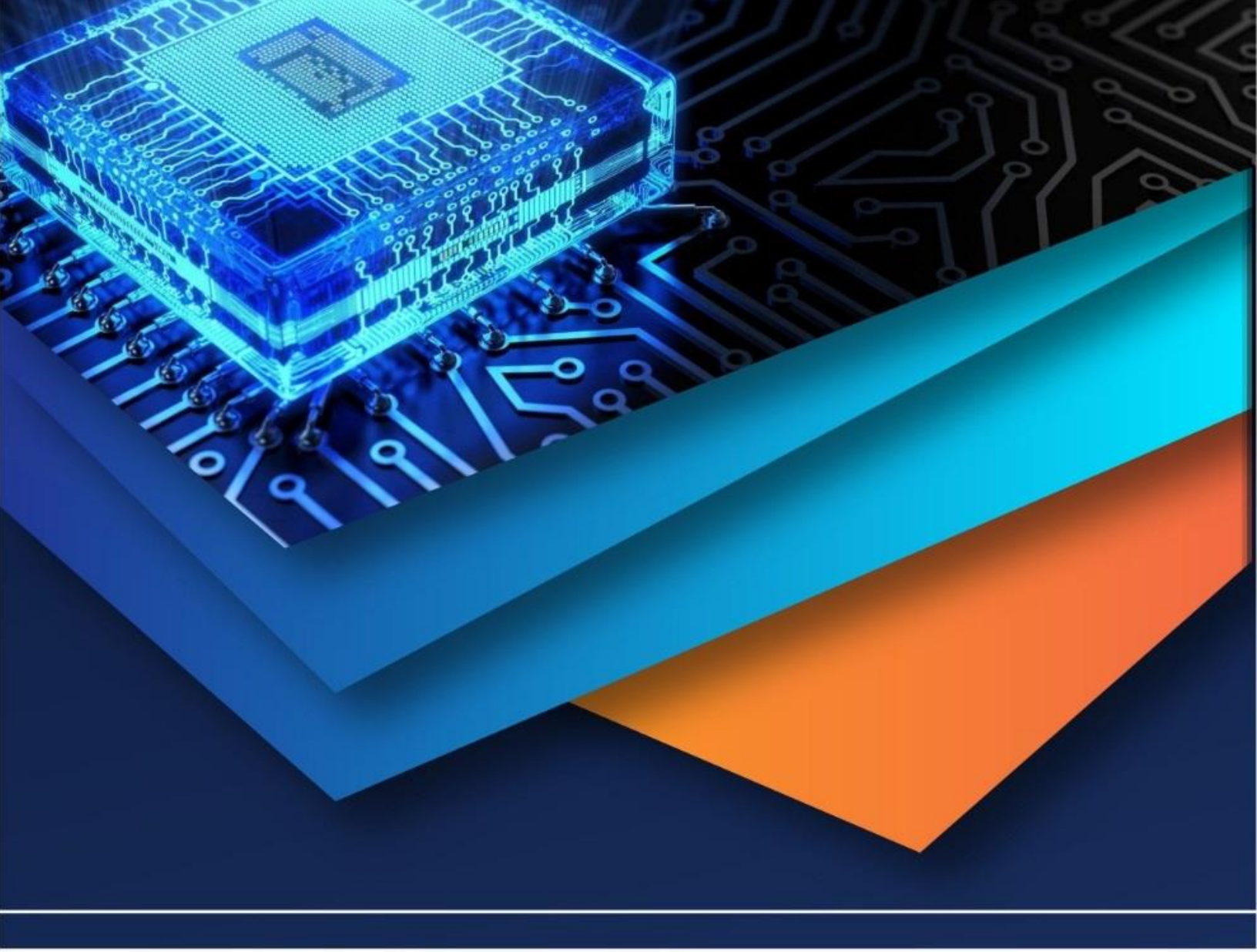
This paper has presented the Cloud Honeypot Monitoring System, a comprehensive platform for attracting, capturing, analyzing, and visualizing malicious activities targeting cloud- based services. The system was designed to address a fundamental gap in cloud security research: the absence of a safe, isolated, and fully instrumented environment in which attacker behavior canbeobservedwithout constrainingthedepth of instrumentation or risking production infrastructure.

By deploying multi-protocol honeypots on AWS EC2 within an isolated Virtual Private Cloud, integrating CloudWatch and S3 for real-time log capture and durable storage, implementing a Python Flask backend providing log parsing, event correlation, signature matching, and anomaly detection, and delivering a React and D3.js dashboard for interactivevisualization,thesystemachievesitsstatedresearch objectivesinfull.Themathematicalmodelpresentedprovidesa rigorous formal specification defining event extraction functions, honeypot state transition models, correlation windows, detection predicates, anomaly score functions, and alert generation logic.

ControlledattacksimulationsusingNmap,Metasploit,and custom Python scripts validated the end-to-end pipeline, demonstrating correct capture and classification of SSH brute force, credential harvesting, SQL injection, path traversal, and web application attack activities. The modular architecture ensures the system can serve as a reusable research substrate, withindividualcomponentsreplaceableandextensibleasfuture requirements evolve. The Cloud Honeypot Monitoring System contributes a validated, replicable, and extensible platform for advancingempiricalunderstandingofcloud-targetedadversarial behavior and for developing the evidence-based defensive strategies that modern cloud security demands.

REFERENCES

[1] P. A. Panditre and V. B. Gaikwad, “Attack detection in cloudvirtual environment and prevention using honeypot,” in Proc. IEEEInt. Conf. Comput. Commun. Informatics (ICCCI), 2018.
[2] Y. Uchibori, T. Sato, K. Nagayoshi, and K. Sakurai, “Honeypotmethod to lure attackers without holding crypto-assets,” in Proc.IEEE Int. Conf. Blockchain Cryptocurrency (ICBC), 2024.
[3] J. Franco, A. Aris, B. Canberk, and A. S. Uluagac, “A survey ofhoneypotsandhoneynetsforInternetofThings,IndustrialInternetof Things, and cyber-physical systems,” IEEE Commun. SurveysTuts., vol. 23, no. 2, pp. 997–1017, Secondquart. 2021.
[4] M. Nawrocki, M. Wählisch, T. C. Schmidt, C. Keil, and J.Schönfelder, “A survey on honeypot software and data analysis,”arXiv preprint arXiv:1608.06249, 2016.
[5] S. Provos and T. Holz, Virtual Honeypots: From Botnet TrackingtoIntrusionDetection.UpperSaddleRiver,NJ:Addison-Wesley,2007.
[6] Amazon Web Services, “Amazon CloudWatch documentation,”AWS Documentation, 2024. [Online]. Available:https://docs.aws.amazon.com/cloudwatch/
[7] Amazon Web Services, “Amazon EC2 documentation,” AWSDocumentation, 2024. [Online]. Available:https://docs.aws.amazon.com/ec2/
[8] Amazon Web Services, “Amazon S3 documentation,” AWSDocumentation, 2024. [Online]. Available:https://docs.aws.amazon.com/s3/
[9] M.Armbrustetal.,“Aviewofcloudcomputing,”Commun.ACM,vol. 53, no. 4, pp. 50–58, Apr. 2010.
[10] H. Burch and W. Cheswick, “Tracing anonymous packets to theirapproximate source,” in Proc. USENIX Large Installation Sys.Admin. Conf. (LISA), 2000, pp. 319–327.
[11] TheMetasploitProject,“Metasploitframeworkdocumentation,”
[12] Rapid7,2024.[Online].Available:https://docs.metasploit.com/
[13] G. Lyon, Nmap Network Scanning: The Official Nmap ProjectGuide to Network Discovery and Security Scanning. Insecure.ComLLC,2009.
[14] R.McGrewandR.Vaughn,“Experienceswithhoneypotsystems:Development, deployment, and analysis,” in Proc. 39th Annu.Hawaii Int. Conf. Syst. Sci. (HICSS), 2006.
[15] M.FiccoandM.Rak,“Stealthydenial-of-servicestrategyincloudcomputing,”IEEETrans.CloudComput.,vol.3,no.1,pp.80–94,Jan.–Mar.2015.
[16] G.Loukas,T.Vuong,R.Heartfield,G.Sakellari,Y.Yoon,andD.Gan,“Cloud-basedcyber-physicalintrusiondetectionforvehiclesusing deep learning,” IEEE Access, vol. 6, pp. 3491–3508, 2018.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY