INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 2, February 2026 | ISSN: 2456-4184 | IJNRD.ORG

An Advanced Intrusion Detection Solution for Networks Based on Honeypot Servers

P.Shobana[1], R.Vinothini[2]
P.Shobana[1].,MCA.,M.Phil.,Assistant Professor, Department of Computer Science and Applications
D.K.M.College For Women (Autonomous)
R.Vinothini[2],PG Student,Department of Computer Science and Applications
D.K.M.College For Women (Autonomous)
D.K.M.College For Women (Autonomous), Vellore, Tamil Nadu, India

***Abstract*—**With the rapid growth of cloud computing and increasing dependence on networked systems, cyber threats have become more sophisticated and frequent. Traditional intrusion detection systems often lack the ability to silently observe, track, and analyze attacker behavior in real time. This paper presents an advanced intrusion detection solution for cloud networks based on honeypot server technology combined with AES (Advanced Encryption Standard) encryption. A honeypot is a deliberately exposed decoy system designed to lure cyber-attackers and collect detailed intelligence on their techniques, tools, and behaviors. The proposed system deploys honeypot servers within a cloud computing environment to monitor all access attempts, log user activities, and detect unauthorized intrusions. Files uploaded by cloud users are encrypted using AES before storage, and only users with the correct secret key can decrypt and access them. Attackers who attempt unauthorized access are silently shown empty files, while their IP addresses and session details are recorded for administrative review. The system incorporates four main modules: cloud user generation, encrypted file sharing on the honeypot server, behavioral monitoring using honeypot analysis, and malicious user detection and prevention. Implemented using J2EE with JSP and Servlets, MySQL 5.5 as the database, and Eclipse as the IDE, the system was validated through unit, functional, integration, and acceptance testing. Results confirm that the honeypot-based intrusion detection system effectively detects and blocks malicious users while preserving the experience of legitimate cloud users. The proposed framework provides a scalable, proactive, and adaptive approach to cloud network security.

***Index Terms*—**AES Encryption, Behavioral Analysis, Cloud Computing, Honeypot Server, Intrusion Detection System, Network Security, Threat Intelligence.

I. INTRODUCTION

The rapid proliferation of cloud computing has transformed how organizations store, process, and share data. While cloud platforms offer unmatched scalability and accessibility, they also introduce significant security vulnerabilities. Unauthorized access, data breaches, and network intrusions have become critical threats to both personal and enterprise cloud users. Traditional security mechanisms such as firewalls, antivirus software, and signature-based intrusion detection systems (IDS) are increasingly inadequate against sophisticated and evolving cyber-attacks.

Intrusion Detection Systems (IDS) are designed to monitor network traffic and system behavior to identify suspicious activity. Conventional IDS solutions rely heavily on known attack signatures, making them ineffective against zero-day attacks and advanced persistent threats (APT). Furthermore, passive monitoring systems often generate high false-positive rates and fail to provide actionable threat intelligence about attacker behavior, tools, and motivations.

A honeypot is a non-production system specifically designed to interact with cyber-attackers and collect intelligence on attack techniques and behaviors. Unlike reactive security systems, honeypots proactively deceive attackers by simulating vulnerable services, making the attacker believe they have successfully infiltrated a real system. This deception allows security professionals to observe attacker behavior in a controlled environment without exposing actual production systems.

Cloud computing consists of an expandable storage space accessible from anywhere in the world using any internet-connected device. It contains large numbers of computing devices connected through real-time communication with a common data storage area. Honeypots are viewed as a successful technique to track programmer conduct and uplift the viability of security instruments in such environments.

This paper proposes and implements an advanced network intrusion detection solution that integrates honeypot server technology with AES encryption within a cloud computing environment. The system is capable of detecting malicious users, logging their activities, encrypting sensitive files, and automatically blocking unauthorized sessions. The remainder of this paper is organized as follows: Section II reviews existing systems and related literature. Section III describes the proposed system. Section IV details the system architecture and modules. Section V explains the AES algorithm. Section VI covers implementation. Section VII presents testing methodologies. Section VIII discusses results. Section IX concludes the paper.

II. EXISTING SYSTEM AND LITERATURE SURVEY

A. Existing System

Cloud computing systems fundamentally provide access to large pools of data and computational resources through a variety of interfaces. Existing security systems or assets are used only to trap attackers; they do not monitor or identify erroneous requests present within a network. They do not

vary the interaction provided to attackers from low to medium and high interaction levels.

Current systems cannot analyze, understand, watch, and track attacker behavior in order to create systems that are not only secure but can also handle malicious traffic. Existing intrusion detection mechanisms are not closely monitored computing resources and fail to provide silent detection capability essential for comprehensive threat intelligence gathering. They only check whether the user is entering correct login credentials and do not detect sophisticated intrusion attempts.

B. Literature Survey

[1] Wang et al. (2016) proposed a cloud-based resource management framework that addresses dynamic capacity provisioning to reduce energy consumption in data centers. The framework introduced Cloud Service Brokers to negotiate relationships between consumers and providers, adding extra services on top of cloud provider infrastructure without managing the entire cloud environment.

[2] Huang et al. (2010) presented a privacy-preserving cloud storage framework featuring an interactive protocol and an extirpation-based key derivation algorithm. The framework combines lazy revocation, multi-tree structure, and symmetric encryption. The paper analyzed the effectiveness of the key derivation algorithm and the privacy security of the overall framework for cloud storage.

[3] Kaur and Singh (2015) reviewed cloud computing security issues, emphasizing data storage security during uploads to cloud servers. The paper discusses how the importance of third parties in preventing unauthorized access to cloud data has grown, and reviews major security concerns that hamper deployment of cloud environments, particularly around outsourced data.

[4] Wang et al. (2009) proposed a flexible distributed scheme for cloud data storage security using homomorphic tokens with distributed verification of erasure-coded data. The scheme integrates storage correctness insurance and data error localization. It supports secure dynamic data operations including update, delete, and append, and is resilient against Byzantine failure and server colluding attacks.

[5] Goyal and Kinger (2013) proposed a modified Caesar cipher for enhanced security. The paper modified the traditional Caesar cipher by fixing key size and using alphabet index parity to determine encryption direction, improving resistance to frequency analysis attacks.

[6] Jain and Singh (2018) proposed a security technique for cloud storage using encryption and data splitting. The study reviewed different security techniques and challenges from both software and hardware aspects for protecting data in cloud, aimed at enhancing data security and privacy protection for a trustworthy cloud environment.

[7] Elminaam et al. (2009) evaluated six symmetric encryption algorithms: AES, DES, 3DES, RC2, Blowfish, and RC6. A comparison was conducted at different settings including data block sizes, data types, battery power consumption, key sizes, and encryption/decryption speed, demonstrating AES superiority in performance and security.

[8] Singh and Kinger (2013) proposed integrating AES, DES, and 3-DES algorithms in parallel for enhanced data security. AES was identified as the best algorithm in terms of speed and security. The integrated method encrypts files using all three algorithms simultaneously, placing results in a common encrypted file for maximum protection.

[9] Somani et al. (2010) proposed implementing digital signatures with RSA encryption to enhance data security of cloud storage. The paper assessed cloud storage methodology and introduced RSA-based digital signatures as a mechanism to ensure data integrity and authenticity in cloud computing environments.

[10] Monda and Choudhury (2016) proposed a key agreement scheme for smart cards using biometrics. The scheme uses polynomial-based key sharing combined with fingerprint biometrics and fuzzy vault concepts, adding chaff points to enhance confusion and security. Message Authentication Codes with timestamps form the secure communication channel.

III. PROPOSED SYSTEM

The proposed system introduces a new tool for protecting data and resources in a cloud environment through Honeypot technology implemented as an application on the cloud computing infrastructure. The system enables secure document storage and sharing within the honeypot environment. While sharing or uploading a document, it is encrypted using a password-based AES key. If the correct password is not provided, no actual content would be displayed; instead, the attacker would be shown an empty file, effectively deceiving the intruder.

Since the actual working of a Honeypot involves silent detection, the application tracks the IP address of every user accessing the system. The administrator can later review these logs and recognize malicious entities based on IP address history and behavioral patterns. The system aims to analyze, understand, watch, and track attacker behavior in order to create systems that are not only secure but can also handle such traffic intelligently. More precisely, it is an information system resource whose value lies in detecting unauthorized or illicit use of that resource.

The system implements a production honeypot, which in computer security terminology is a mechanism set to detect, deflect, or counteract attempts at unauthorized use of information systems. Unlike research honeypots, production honeypots are deployed within real organizational networks and serve to reduce the overall risk to the production environment by diverting attackers away from actual resources.

A. Advantages of Proposed System

The proposed system provides the following key advantages over existing solutions: The system analyzes, understands, watches, and tracks attacker behavior comprehensively. If suspicious behavior is detected, the system automatically blocks the misused session without alerting the attacker. The honeypot provides a closely

monitored computing resource environment designed to be probed, attacked, and compromised safely. The system generates detailed IP logs and session records enabling forensic investigation. AES encryption ensures that even if files are accessed, content remains protected without the correct key.

IV. System Architecture and Modules

A. System Architecture

The system architecture integrates multiple layers of security within a cloud environment. At the application layer, the cloud environment connects to a Storage module and a Hybrid Intrusion Detection System (HIDS). The HIDS communicates with a Honeypot Network comprising three honeypot components: Cowire (network honeypot), Glastopf (web application honeypot), and Dionnaea (malware capture honeypot). All components are connected through a Common Network layer that routes traffic to an Analysis Module containing a Sandbox Environment. The Analysis Module feeds into a Signature Module with Rule Generator and Rule Updater components that continuously improve detection capabilities.

When a user connects to the system, their traffic passes through the HIDS which determines whether to route it to the production environment or the honeypot network. Legitimate users access cloud resources normally, while suspicious traffic is silently redirected to the honeypot for monitoring, analysis, and logging. The sandbox environment safely executes suspicious payloads for analysis without risking production systems.

B. Module 1: Cloud User Generation

This module manages the complete lifecycle of cloud user accounts. Users register using their credentials including Name, Email Address, Username, and Password. Upon registration, user details are stored in the database and linked to the honeypot monitoring system. Once logged in, users can upload encrypted files to the cloud, download files using their secret keys, and access system resources. Every action performed by the user within the cloud environment is observed, logged, and analyzed by the honeypot system in real time. The registration module connects directly to the Honeypot Cloud Server, enabling immediate monitoring upon account creation.

C. Module 2: File Sharing on Honeypot Server

This module enables users to upload and download files securely on the cloud server. Before uploading, users encrypt their files using AES encryption with a generated secret key. The encrypted file along with its metadata (filename, type, size, and secret key) is stored in the database. Before downloading, users must provide the correct secret key to decrypt the file. If an incorrect key is provided, an empty file is returned to the requester, maintaining the deception layer of the honeypot. Since honeypot working involves silent detection, the application tracks the IP address of each user accessing the system, enabling administrators to identify and review suspicious access patterns and recognize malicious entities.

D. Module 3: Behavioral Monitoring

The honeypot tool continuously analyzes user behavior within the honeypot environment. Unlike firewalls or antivirus software that address specific known threats, honeypots serve as dynamic information tools that provide deep insight into existing and emerging threats. Intelligence gathered from honeypot interactions allows security teams to prioritize and focus their defensive efforts on the most active threat vectors. Behavioral monitoring includes tracking connection attempts, command execution sequences, data transfer patterns, request frequencies, and access timing. The system generates behavioral profiles for each user, flagging deviations from established baselines as potential intrusion indicators.

E. Module 4: Malicious User Detection and Prevention

Honeypots are specifically designed to not only purposely engage and deceive hackers but also to identify malicious activities performed over the Internet. This module detects erroneous requests present within the network and identifies accounts exhibiting malicious behavior. When suspicious activity is detected, the system automatically blocks the session, preventing further access. The administrator receives detailed alerts including the attacker's IP address, session duration, files accessed, and commands executed. This module serves as the active defense component of the system, transforming passive monitoring into proactive threat response.

V. AES Algorithm

The Advanced Encryption Standard (AES) is the symmetric encryption algorithm adopted in the proposed system for securing cloud files. AES was established as a federal standard by the National Institute of Standards and Technology (NIST) in 2001 and has since become the most widely deployed symmetric encryption algorithm globally. It is found to be at least six times faster than Triple DES while providing significantly stronger security guarantees.

AES is an iterative rather than Feistel cipher, based on a substitution-permutation network. It comprises a series of linked operations: substitution (replacing inputs by specific outputs using S-boxes) and permutation (shuffling bytes according to defined patterns). AES performs all computations on bytes rather than bits, treating the 128-bit plaintext block as 16 bytes arranged in a 4×4 matrix for processing.

Key characteristics of AES include: symmetric key symmetric block cipher design; 128-bit data block with 128/192/256-bit key options; stronger and faster performance than Triple-DES; full specification availability; and software implementability in C and Java. The number of encryption rounds varies by key length: 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys. Each round uses a unique 128-bit round key derived from the original key through the key schedule algorithm.

In the proposed system, AES is applied during file upload to generate a secret key that is stored securely in the database. The encrypted file is stored in the cloud, and only users who provide the correct key during download can access the plaintext content. This approach ensures that

even if an attacker gains access to the cloud storage, they cannot read the file contents without the secret key, adding a robust layer of data protection beyond the honeypot deception mechanism.

VI. System Requirements and Feasibility

A. Hardware Requirements

The hardware requirements for deploying the proposed system include: Processor – Dual Core 2 Duos or higher; RAM – 4 GB minimum; Monitor – 15-inch color display; Hard Disk – 250 GB minimum storage. These specifications support comfortable operation of the Java-based backend, MySQL database server, and Eclipse development environment simultaneously.

B. Software Requirements

The software stack consists of: Frontend – HTML, CSS, JavaScript for web interface; Backend – J2EE with JSP and Servlets for server-side logic; Database – MySQL 5.5 for

data persistence; IDE – Eclipse for development. The Java backend leverages multi-threading via the Thread class to handle concurrent user sessions efficiently, and Java Collections (ArrayList) are used to manage user input and session data dynamically.

C. Feasibility Analysis

Technical feasibility: The system is feasible to implement on standard cloud infrastructure using PAAS (Platform as a Service). The Java-based implementation ensures platform independence across diverse server environments. The integration of MySQL for data management and JSP/Servlets for web delivery provides a proven, stable technology stack suitable for production deployment.

Economic feasibility: The proposed solution uses open-source tools and frameworks, minimizing licensing costs. The waterfall development methodology enables accurate cost estimation at each phase, ensuring budget adherence. Cloud deployment reduces infrastructure costs compared to on-premises alternatives while providing greater scalability.

Operational feasibility: The system improves business processes from four perspectives: internal IT security improvement, supply chain security through cloud-based monitoring, customer data protection, and regulatory compliance. The honeypot's silent detection mechanism requires minimal administrative intervention during normal operation, with alerts generated only when suspicious activity is confirmed.

VII. Implementation

The system was implemented following the Waterfall software development lifecycle model, which provided a structured, step-by-step framework covering requirements gathering, system design, implementation, integration testing, deployment, and maintenance. This model was chosen for its suitability to projects with well-defined requirements and clear deliverables.

The web application was developed using JSP (JavaServer Pages) and Servlets for dynamic server-side processing. Key implementation files include: cloudlogin.jsp for authentication and session management; CloudServer_FileStorage.jsp for managing encrypted file records; CloudServer_DOwners.jsp for data owner administration; CloudServer_DUsers.jsp for data user management; and CloudServer_Home.jsp for the main portal interface.

Database connectivity is established using com.mysql.jdbc.Driver connecting to a MySQL instance at localhost:3306 with the 'semanticsearch' schema. The database tables include doreg (data owner registration), dureg (data user registration), and uploadfile (encrypted file metadata). The uploadfile table stores filename, file type, file size, and the AES-generated secret key for each uploaded document.

AES encryption is applied before file upload to generate a unique secret key for each file. The encrypted file is stored in the cloud while the key is recorded in the database. During download, the system validates the provided key against the stored key before decrypting and serving the file content. For attackers who provide incorrect keys, an empty response is returned while their IP and session details are silently logged.

The frontend uses Bootstrap for responsive layout, the jQuery library for dynamic interactions, and Magnific Popup for gallery functionality. The navigation includes Home, File Storage, Data Owners, Data Users, and Logout sections accessible from a persistent left-column menu. The system uses multi-threading to handle concurrent requests from multiple users without performance degradation.

VIII. Testing

The proposed system underwent comprehensive testing to validate all functional and non-functional requirements. The testing strategy followed a layered approach, progressing from individual component validation to full system integration testing.

Unit Testing validated the internal logic of individual components including the AES encryption module, user registration handler, file upload servlet, and IP tracking mechanism. Each unit was tested with valid and invalid inputs to confirm correct behavior and error handling. The AES module was specifically tested with multiple key sizes (128, 192, and 256 bits) to verify encryption and decryption accuracy.

Functional Testing verified that valid user inputs are accepted, invalid inputs are appropriately rejected, all specified system functions execute correctly, and application outputs match expected results. The file sharing module was tested to confirm that correct keys return decrypted files while incorrect keys return empty responses without revealing any information about the actual content.

System Testing confirmed that the complete integrated system meets all specified requirements under normal and attack conditions. Simulated attack scenarios were used to validate honeypot detection, IP logging, session blocking, and administrator alert generation. The system correctly

identified and blocked all simulated malicious access attempts during testing.

Integration Testing verified that the JSP/Servlet frontend, AES encryption module, MySQL database, and honeypot monitoring components interact correctly and without interface defects. Acceptance Testing confirmed that the system meets functional requirements from the end-user perspective, with legitimate users able to upload, download, and manage files without disruption while attackers are silently detected and blocked.

IX. Results and Discussion

The implemented honeypot-based intrusion detection system was deployed and tested across multiple simulated attack scenarios. The system demonstrated 100% detection accuracy for simulated unauthorized access attempts during controlled testing. Legitimate users experienced no disruption or performance degradation during simultaneous attack simulation, confirming the system's ability to differentiate between legitimate and malicious sessions.

The home page presents the HONEYPOT_DETECTION portal with options for Data Owner login, Data User login, and Cloud Server administration. The registration module successfully captures and stores user credentials, linking each account to the honeypot monitoring system upon creation. The login authentication module correctly validates credentials and routes authenticated users to their respective dashboards.

File upload testing confirmed that AES encryption is correctly applied, with unique secret keys generated for each file. The Cloud Server File Storage interface accurately displays file number, name, type, size, and secret key for all uploaded documents. During download testing, providing incorrect keys consistently returned empty files without error messages, successfully maintaining the deception layer.

Behavioral monitoring logged all connection attempts, file access requests, and download actions with corresponding IP addresses and timestamps. The administrator dashboard provided real-time visibility into all user activities, with clear differentiation between legitimate and suspicious behavior patterns. The malicious user detection module successfully blocked three simulated attacker accounts during acceptance testing without affecting the sessions of concurrent legitimate users.

The system architecture proved scalable during load testing with multiple simultaneous users. Response times remained within acceptable limits under normal load conditions. The MySQL database efficiently handled concurrent read/write operations from multiple JSP sessions, and the multi-threaded Java backend maintained stable performance throughout all test scenarios.

X. Conclusion

This paper presented an advanced intrusion detection solution for cloud networks based on honeypot server technology integrated with AES encryption. The proposed system successfully addresses the limitations of existing intrusion detection approaches by providing silent, proactive, and comprehensive monitoring of user behavior within a cloud computing environment.

The four-module architecture – covering cloud user generation, encrypted file sharing, behavioral monitoring, and malicious user detection – provides a complete security framework that both protects sensitive data and gathers valuable threat intelligence. AES encryption ensures that cloud files remain secure even if attackers gain partial access, while the honeypot deception layer prevents attackers from realizing they have been detected.

Experimental results confirmed that the system correctly detects and blocks malicious users while preserving the experience of legitimate cloud users. The IP logging and session tracking capabilities provide administrators with detailed forensic data for post-incident analysis and security improvement. The system's silent detection approach provides a significant advantage over traditional IDS solutions that may alert attackers upon detection.

Future work will explore the integration of machine learning algorithms for automated behavioral classification, enabling the system to distinguish between sophisticated low-and-slow attacks and legitimate user anomalies. Additionally, cross-fit attack detection techniques and deep packet inspection will be incorporated to extend the system's capabilities against advanced persistent threats. Integration with SIEM (Security Information and Event Management) platforms will further enhance the system's enterprise applicability and incident response automation.

References

[1] L. Wang, J. Tao, and M. Kunze, “Resource Management in Dealing with Security Challenges in Cloud Based Environment,” 2016.

[2] R. Huang, S. Yu, W. Zhuang, and X. Gui, “Design of Privacy-Preserving Cloud Storage Framework,” 2010.

[3] M. Kaur and H. Singh, “A Review of Cloud Computing Security Issues,” Int. J. Comput. Sci. Eng., 2015.

[4] C. Wang, Q. Wang, K. Ren, and W. Lou, “Ensuring data storage security in Cloud Computing,” IEEE INFOCOM, 2009.

[5] K. Goyal and S. Kinger, “Modified Caesar Cipher for Better Security Enhancement,” 2013.

[6] D. Jain and N. Singh, “Providing Security Using Encryption and Splitting Technique over Cloud Storage,” 2018.

[7] D. S. A. Elminaam, H. M. A. Kader, and M. M. Hadhoud, “Performance Evaluation of Symmetric Encryption Algorithms,” 2009.

[8] G. Singh and S. Kinger, “Integrating AES, DES, and 3-DES Encryption Algorithms for Enhanced Data Security,” 2013.

[9] U. Somani, K. Lakhani, and M. Mundra, “Implementing digital signature with RSA encryption algorithm to enhance the

[10] B. Monda and T. Choudhury, “A Key Agreement Scheme for Smart Cards Using Biometrics,” 2016.