iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.79988

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 IV Apr 2026- Available at www.ijraset.com

System Immune: An AI-Driven Self-Healing Chaos Engineering Platform

Ms. Surabhi Chavhan[1], Nishad Palsapure[2], Mohd Ziyafat Abbas[3], Manas Bhajgawre[4], Mayur Chaple[5], Mohit Mene[6], Manish Khalsinghe[7]
Department of Computer Science, G.H. Raisoni University

Abstract: ***Modern distributed cloud systems are inherently complex, making them highly susceptible to failures such as service crashes, network latency, and database unavailability. Traditional recovery mechanisms depend heavily on manual intervention, resulting in increased downtime and operational costs. This paper proposes System Immune, an intelligent self-healing platform that combines Chaos Engineering with Reinforcement Learning to autonomously detect, diagnose, and recover from failures. Inspired by industry tools like Chaos.***
Monkey, the system introduces controlled fault injection while leveraging Q-Learning to determine optimal recovery strategies. The architecture utilizes Docker-based containerization for realistic failure simulation and a Flask-based backend for monitoring. Experimental evaluation demonstrates significant improvements in system resilience, reduced recovery time, and adaptive learning over repeated failure scenarios.The proposed system represents a step toward fully autonomous cloud infrastructure management.
Keywords: Chaos Engineering, Reinforcement Learning , Self-Healing Systems , Cloud Infrastructure Resilience , Q-Learning.

I. INTRODUCTION

Cloud-native systems have revolutionized how applications are deployed and scaled. However, with increasing distribution and interdependencies among services, system reliability has become a major challenge. Failures in modern systems are not exceptions but expected events due to hardware issues, software bugs, and unpredictable network conditions.
Traditional approaches to system reliability rely on monitoring tools that alert engineers when failures occur. While effective for detection, these systems still require human intervention for diagnosis and recovery. This introduces delays and increases downtime, especially in large-scale systems where identifying the root cause is complex. Chaos Engineering emerged as a proactive solution to test system resilience by intentionally introducing failures. Tools like Chaos Monkey simulate real-world disruptions by randomly terminating services. While this improves robustness, it lacks automated recovery mechanisms. To address these limitations, this paper introduces *System Immune*, a platform that integrates Chaos Engineering with Artificial Intelligence. The system mimics a biological immune system by continuously monitoring, detecting anomalies, and autonomously healing failures. By incorporating Reinforcement Learning, the system improves its recovery strategies over time, reducing reliance on human operators.

II. RELATED WORK

A. *Chaos Engineering*

Chaos Engineering focuses on improving system resilience by injecting controlled failures. Chaos Monkey is one of the earliest tools used in production environments. It randomly shuts down instances to ensure systems can withstand unexpected failures.
However, Chaos Engineering tools primarily focus on failure injection rather than recovery, leaving a gap in autonomous resilience.

B. *Reinforcement Learning in Distributed Systems*

Reinforcement Learning (RL) enables systems to learn optimal decisions through interaction with the environment. Q-Learning, a widely used RL algorithm, allows agents to map system states to actions based on rewards.
In distributed systems, RL has been applied to:

1) Resource allocation
2) Load balancing
3) Fault recovery

Despite its advantages, RL is rarely integrated with real-world infrastructure for automated system healing.

©IJRASET: All Rights are Reserved

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

C. Security and Reliability Standards

The OWASP Top 10 highlights critical vulnerabilities in modern applications. While focused on security, it emphasizes the importance of system robustness and fault tolerance.

D. Research Gap

Existing research shows:

1) Chaos Engineering lacks intelligence
2) RL lacks infrastructure integration
3) Limited work combining both

System Immune bridges this gap by integrating AI with real-world failure simulation.

III. SYSTEM ARCHITECTURE

System Immune is designed as a modular architecture consisting of four key layers:

1) Infrastructure Layer

The system uses Docker containers to isolate services such as the backend and database. This enables:

- Controlled failure injection
- Easy scalability
- Realistic simulation of production environments

2) Monitoring Layer

The monitoring system continuously checks service health through API endpoints. It detects anomalies such as:

- Service downtime
- Database disconnection
- Increased response latency

3) AI Decision Layer

The AI agent uses Q-Learning to determine the best recovery action. It maintains a Q-Table where:

- Rows represent system states
- Columns represent possible actions

Example:

- State: Database Down
- Actions: Restart DB, Restart Backend

4) Execution Layer

Once a decision is made, recovery actions are executed using Docker commands. This ensures:

- Fast response
- Minimal manual intervention

5) User Interface

A dashboard provides:

- Real-time monitoring
- Manual chaos injection

IV. ATTACK GENERATION AND DETECTION STRATEGY

The system utilizes a comprehensive attack generation module that produces payloads across multiple vulnerability categories. Each category is associated with a specific detection mechanism to ensure accurate identification of security flaws.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Attack Category	Payload Count	Detection Method
XSS (Reflected)	6	Response body analysis, script tag detection
SQL Injection (Error-Based)	6	Database error string matching
SQL Injection (Time-Based)	4	Response delay analysis (>5s threshold)
Command Injection	8	OS command output patterns in response
LDAP Injection	4	LDAP error message detection
Path Traversal	5	File content patterns (e.g., /etc/passwd)
Open Redirect	5	HTTP 3xx redirect destination analysis
SSRF	4	Internal resource access detection
XXE	2	XML parser error and file content detection

This multi-layered approach enables the system to detect a wide range of vulnerabilities using pattern-based, behavioral, and timing-based techniques.

A. *Workflow*

The system follows a continuous feedback loop:

1) Monitoring – System health is continuously tracked
2) Detection – Failure is identified
3) Diagnosis – AI determines system state
4) Decision – Best action selected using Q-table
5) Execution – Recovery action applied
6) Validation – System health verified
7) Learning – Q-table updated based on outcome

This loop enables continuous improvement and adaptability.

V. REINFORCEMENT LEARNING MODEL

A. *Q-Learning Framework*

The AI agent learns using:

$$Q(s,a) = Q(s,a) + \alpha[R + \gamma max\, Q(s',a') - Q(s,a)]$$

Where:

- s: current state
- a: action
- R: reward
- α: learning rate
- γ: discount factor

B. *Reward Mechanism*

- Successful recovery → +10
- Failed recovery → -1

C. *Learning Behavior*

Over time, the system:

- Learns optimal recovery strategies
- Reduces incorrect actions
- Improves response efficiency

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

VI. IMPLEMENTATION

The system is implemented using:

1) Backend: Python Flask
2) Database: PostgreSQL
3) Containerization: Docker
4) AI Module: Q-Learning

The modular design ensures easy deployment and scalability.

VII. EVALUATION

A. *Test Scenarios*

- Database crash
- Backend failure
- Combined failures

B. *Metrics*

- Recovery time
- Success rate
- Learning efficiency

C. *Results*

The system shows:

- Reduced downtime
- Faster recovery
- Improved adaptability over time

Criterion	System Immune	OWASP ZAP
Setup Complexity	Simple (pip install)	Complex (GUI/API setup)
Scan Speed	< 2 minutes	5-30+ minutes
Remediation Guidance	Detailed, step-by-step	Basic references only
Report Clarity	Owner-friendly HTML	Technical XML/HTML
Deduplication	SHA256 fingerprinting	Basic deduplication
Ethical Consent Check	Built-in (--consent flag)	Not enforced
XSS Detection	Pattern + Playwright verify	Pattern matching only
Open Source	Yes	Yes
Programming Language	Python	Java
Payload Count	50+	200+ (with extensions)

VIII. ADVANTAGES

1) Autonomous recovery
2) Continuous learning
3) Reduced operational cost
4) Scalable architecture

IX. CHALLENGES

1) Initial training phase
2) Limited state representation
3) Dependency on monitoring accuracy.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

X. FUTURE WORK

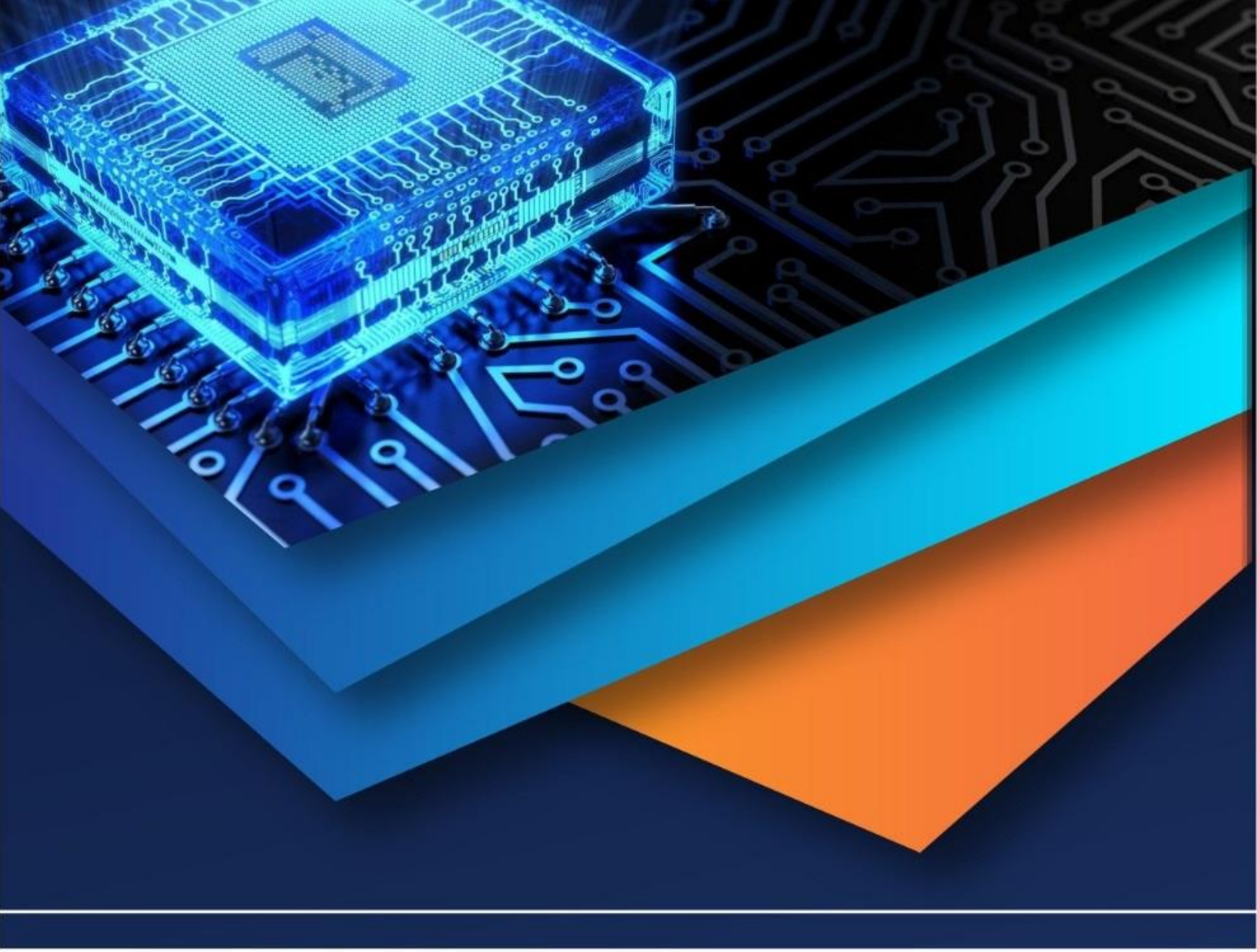
1) Deep Reinforcement Learning
2) Kubernetes integration
3) Predictive failure detection

XI. CONCLUSION

System Immune demonstrates how AI can enhance system resilience by enabling autonomous recovery. By combining Chaos Engineering with Reinforcement Learning, the system not only tests failures but also learns to fix them efficiently. This approach represents the future of intelligent cloud infrastructure.

REFERENCES

[1] Chaos Engineering Literature
[2] IEEE Research Papers
[3] OWASP Top 10
[4] Reinforcement Learning Textbooks
[5] Docker Documentation



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)