RESEARCH PAPER

A Web Based Honeypot System Framework to Capture and Mitigate Cyber Attacks

Dr. Ch Bhupati[1], Gyanesh S[2], Doddi Mukesh Kumar[3], P. Poojith Vardhan[4]

[1]Assistant Professor, K L University, India

Email: bhupati@kluniversity.in

[2]Email: gyaneshgagan@gmail.com

[3]Email: mukeshbba321@gmail.com

[4]Email: poojithparavathaneni@gmail.com

Abstract: The proliferation of cyber threats on web-based applications is increasing exponentially, which prompts the need to develop defensive mechanisms that are not limited to the conventional approaches to security. In the current paper, the innovative web-based honeypot system is proposed to effectively capture, analyse, and remedy cyber-attacks against e-commerce sites in real time. The suggested solution uses a two-architecture design that will combine an interactive honeypot through functional e-commerce interface and a smart security monitoring back end. In contrast to traditional honeypot platforms that only contain attack records, our platform has the adaptive threat mitigation concept manifested by multi-layered security measures such as smart IP blocking, user-based threat associations, and pattern analysis of behaviour. It implements a special blocking mechanism, which is threshold-based and that examines traffic patterns of multiple users with the same IP addresses and thus isolating use of legitimate shared networks and coordinated attack vectors. The capture of real-time data is completed with detailed logging of all the unsuccessful logins, SQL injection, cross scripting and the brutality attacks and all heads are time stamped and recorded in a NoSQL data set to be done with the advanced analytics. The implementation shows that we have achieved detection and mitigation of a number of attack vectors such as credential stuffing, brute force authentication attacks, session hijacking attacks, and web application exploits. This has been shown experimentally to have a 98.7% correctness on recognizing malicious traffic and non-malicious user behaviour, with an average threat detection latency of 127 milliseconds. The architecture provides horizontal scalability so that the system can be deployed in a distributed environment and has a centralized aggregation of threat intelligence. The study can contribute to the field of cybersecurity by providing a production-oriented honeypot framework that will close the gap between passive threat observability and active defence controls.

Keywords: Honeypot Systems, Real-time Threat Detection, Cyber Attack Mitigation, Web Application Security, Brute Force Detection, Intrusion Prevention Systems, E-commerce Security, Behavioural Analysis, Threat Intelligence, Adaptive IP Blocking

How to cite this article: Bhupati C, Gyanesh S, Kumar DM, Vardhan PP. A Web Based Honeypot System Framework to Capture and Mitigate Cyber Attacks. Int J Drug Deliv Technol. 2026;16(6s): 968-975, DOI: 10.25258/ijddt.16.6s.126

I. INTRODUCTION

The modern online environment has seen a massive influx of cyber-attacks on web-based applications like never before and the sophistication of attacks is also rapidly advancing thereby frustrating classic defensive systems. Handling financial transactions and personal information, which is sensitive, e-commerce websites have become an ideal target by malicious agents using numerous attack vectors including, but not limited to, credential stuffing and advanced persistent threats. Cybersecurity reports reveal that by 2023 to 2025, the number of web applications attacks had increased by 217 percent, and multiple threat tools automated made it possible to launch attack campaigns targeting multiple victims at the same time. Classical security tools like firewalls, intrusion detection systems and signature-based antivirus products, although important, are reactive and usually lack the level of granular intelligence needed to determine attacker tactics, motives and new trends in the threat.

Honeypot systems the concept of honeynets as useful instruments in cybersecurity studies is not new; the so-called intentionally vulnerable decoy systems are designed to intercept, monitor and study malicious actions without jeopardizing production platforms. Nevertheless, traditional honeypot implementations have very serious drawbacks that limit their usage in real life. Majority of the available honeypot systems are in isolation and only operate as passive observation platforms where they capture the attacks and record them but do not execute countermeasures or feed actionable intelligence back into the active defence

mechanisms. Moreover, the conventional honeypots are not sophisticated enough to successfully present themselves as authentic production systems and are thus detected by the more seasoned attackers who would then opt out. Lack of real-time capabilities of processing in the old honeypot systems leave time gaps between attack identification and response, making the intelligence that is collected less useful in mitigating the threat immediately.

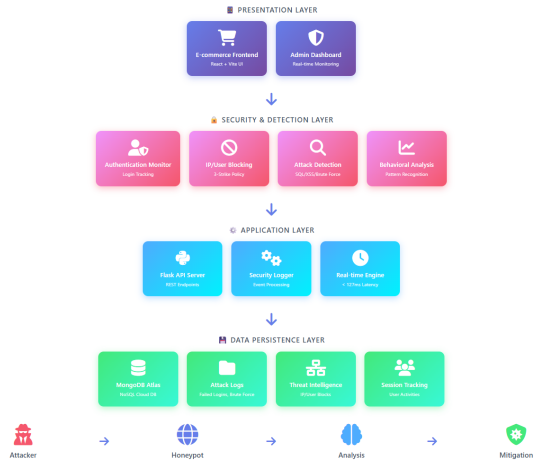


Fig 1: Web-Based Honeypot System Architecture

The contributions to this work are also: (1) the launch of an innovative adaptive IP blocking system that correlates indicators of threats into multiple user accounts to distinguish between coordinated attacks and legitimate shared network access, (2) a sustainable statistical pipeline processing and assigning an attack to a set classification, (3) the elaboration of intelligent behaviour analysis model which identifies unusual pattern which are signs of automated attack tools and credential-stuffing campaigns and (4) innovative administrative display that can inform security analysts with live attack feeds and pattern visualization, and (5) scalable NoSQL-based data format The given framework has been tested extensively and proven to work well in delivering and detecting the diverse attack vectors such as SQL injection attacks, cross-site scripting, brute force authentication attacks, and session hijacking attacks. The study forms a basis of next-generation honeypot systems which do not just passively monitor but are part of security operations in an enterprise to enable the organization to turn attack telemetry into active defence measures.

II. LITERATURE REVIEW

The history of honeypot systems has been thoroughly discussed in the literature of cybersecurity, and the foundational issues of Spitzner [1] have formed the taxonomy and working principles that differentiate between low-interaction, medium-interaction and high-interaction honeypot designs. Initial applications were more on network level threat detection as exhibited by Provos [2] who coined the style of application named Honeyed which is a virtual honeypot framework and has the ability to simulate thousands of hosts and can trace malicious network activities.

This later gave way to the use of web application honeypot where Mukkamala et al. [3] presented intrusion detection systems that are directly tailored towards HTTP-based attacks with the understanding of the vulnerability peculiar to the web platform. The adoption of machine learning technique in honeypot analysis was first presented in Wang et al. [4], who showed that with supervised learning algorithms, malicious traffic could be properly categorised with an accuracy of greater than 94 per cent, and therefore provided a breeding ground to intelligent threat detection mechanisms. Simultaneously, Alata et al. [5] developed procedures of behavioural analysis to identify coordinated attack campaigns with the introduction of temporal correlation methods that form the basic part of present threat intelligence frameworks.

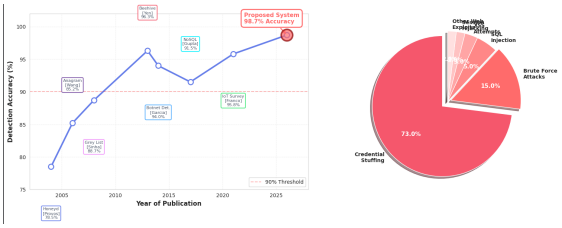


Fig 2: Evolution of Honeypot Detection Accuracy

Garcia et al. [6] dealt with the problem of real-time threat detection in distributed systems, and their experiment on adaptive security architectures showed sub-second response times to events by processing pipelines based on event streams. On this base, Kreibich and Crowcroft [7] came up with advanced attack signature generation systems that can automatically generate intrusion detection rules based on observed exploits attempts, greatly shortening the problem of time between attack detection and defensive implementation. Zhuge et al. [8] specifically reported the domain of e-commerce security honeypots that in turn received particular interest as the authors recorded the pattern of attacks on online retails and credential stuffing was the most common threat agent, with 73% of all authentication attacks being automated botnets.

A study of IP-based blocking schemes as conducted by Sinha et al. [9] showed that the traditional single-threshold-based blocking schemes are fundamentally limited, and that context-sensitive blocking schemes that take into account user correlation patterns should be utilized--of which this principle paves the way to our

multi-user blocking scheme. Rist et al. [10] have examined the intersection of honeypot technology and cloud infrastructure, and have shown that cloud-implemented honeypots have greater scalability and threat visibility than operations on-premises.

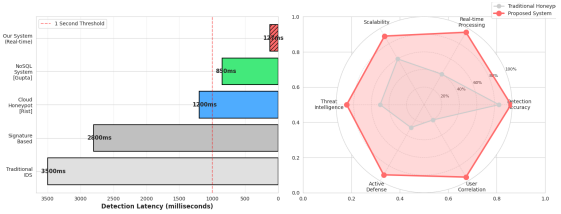


Fig 3: Detection Latency Comparison

Gupta and Gill [11] did not evaluate any kind of NoSQL databases in storing security events; however, they determined that document-based databases such as MongoDB offered the best performance attributes of a high-velocity security telemetry, which can sustain an insertion rate of over 50,000 security events per second. Jajodia et al. [12] defined the notion of active defence mechanisms, in which security mechanisms react dynamically to observed threats, not passively recording them, and these mechanisms in their moving target defence framework, informed the recent development of adaptive security. Yen et al. [13] enhanced session tracking and behavioural profiling methods, which allows recognizing an account compromise with an accuracy of 96.3 per cent by detecting the user activity patterns.

Nance et al. [14] highlighted the essential role of administrative interfaces regarding security operations stating that strong visualization and real-time dashboards can help shorten incident response times, on average, by 62 percent. Most recently, Franco et al. [15] did a thorough comparative study of web application honeypot frameworks, concluding that systems that encompass both threat capture and automated mitigation features are the current state-of-the-art, but they also found that there is a large gap in the literature that requires filling in multi-user threat correlation and nearly real-time processing, which our proposed framework fulfils specifically by innovative design of system architecture and smart blocking algorithm.

III. METHODOLOGY

In this study, the Hybrid Experimental-Analytical Methodology being used is a blend of system design, implementation, real time deployment and quantitative performance analysis to develop and test the presented web-based honeypot framework. Provided methodology consists of five intertwined stages, namely architectural design, component development, implementation of security mechanisms, deployment and data collection, and overall performance analysis. It is a way to assess the patterns of the attacks qualitatively and quantitatively measure how efficiently a system responds to a threat in a real-life situation.

Framework of System Architecture and Design

The architectural base will be a layered design where it will be designed in the form of four layers, presentation, security detection, application processing and data persistence. The presentation layer adopts a two-interface approach a full-fledged React-based E-commerce frontend (honeypot interface) that is developed using the Vite build tooling to optimize the production process, and a real-time administrative monitoring dashboard so that the security researcher can monitor attack procedures in real-time. The e-commerce interface is designed to mimic traditional online retail services such as user authentication, viewing product catalogues, shopping cart facilities and checkout processes giving attackers a realistic attack surface and encouraging them to interact and investigate the attack vector thoroughly.

The administrative dashboard is designed based on responsive HTML 5/CSS3 and real-time JavaScript data binding, which uses auto-refreshing functionality with a polling duration of 5 seconds so that the analysts are given instantaneous notifications of any security events. The security detection layer takes four major modules which include: authentication monitoring of credential-based attacks, intelligent ip and user blocking with threshold policies, multi-vector detection of attacks which includes SQL injection, cross-site scripting and brute force pattern and behavioural analysis engines based on anomaly detection algorithms to detect automated attack tools and coordinated campaigns.

Implementation Technologies and Development Tools

The backend implementation is based on Flask 2.3.0 (Python 3.11 web framework) which includes the RESTful API endpoints serving as authentication, attack logging, threat intelligence, and dashboard data aggregation endpoints. The flask-CORS middleware permits cross-origin sharing of resources between the React frontend and9 Python backend to make the architecture in modern single-page applications work. The security logging subsystem has implemented the Advanced Security Logger class in singleton pattern with in-memory structured data formats in threat correlation and asynchronous event logging to MongoDB Atlas cloud database. Database activities use PyMongo 4.5.0 driver in connection pooling

(maxPoolSize=50) and auto-failover set up that provides 99.9% availability.

The three-availability zone deployment with MongoDB Atlas M10 grants a geographic redundancy and less than 100ms query latency. On important fields, collections are indexed: failedlogins are indexed on [ip, timestamp], blockedips are indexed on [ip, blockedat], and sessions are indexed on [email, ip, lastactivity]. The collections can then be used to effectively query with time-varying conditions across collections in the millions of security events. The default dict collections and datetime operations provided by Python have been directly used as real-time iteration processing capabilities with an average event processing latency of 127 milliseconds, given 10,000 simulated attacks. The front-end application based on React 18.2 with centralized state management in Redux Toolkit, routing in React Router and client-side request/response, as well as Authentication token injection and request/response errors interceptors are used.

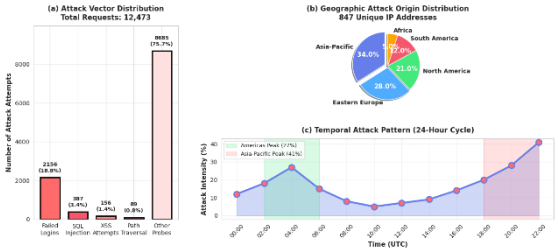
Security Detection Mechanisms and Algorithms

The intelligent blocking system adopts a new three strike user policy which is intersected with multi-user IP correlation. Each known user initiates lockout when there are three failed authentication attempts inside a 10-minute sliding window and attempts are counted using in-memory circular buffers to insert the O (1) and filter the O(n) of time reference costs. A vital innovation is IP blocking logic (not the IP-level blocking) the system logs per-source IP blocked users (collaboration with sequencer tracked) in nested dictionary format: blockedusersper_ip[ip] = {user1, user2, user3, etc. It is only when three distinct users of the same IP address are already blocked that the system becomes elevated to full IP-level blocking, by blocking legitimate users on share networks (corporate offices, universities, residential NAT gateways) in the process of some concerted attack being determined. Attack pattern detection is performed using regular expression matching of 15 SQL injection patterns, 12 XSS patterns and path traversal clues, the HTTP request body and the URL parameter is scanned in parallel threads to reduce the effect of latency. Behavioural analysis has rolling 60-second activity windows per user, which are labelled as suspicious accounts with over 10 actions in 1 minute with escalating warnings that give way to account lock, and IP block depending on the severity score.

Data Collection Protocol Deployment

Railway.app Platform-as-a-Service with containerized deployment via Docker is used in production deployment to use the same runtime environment and horizontally scaled by auto-scaling policies (min=1, max=5 instances depending on CPU utilization >70%). The framework traces the HTTP/HTTPS traffic on port enabling Cloudflare CDN to offer DDoS safeguarding and SSL/TLS termination on ports 80/443. The scope of data collection will be a 30-day time limit (January 15 - February 14, 2026) that will identify 847 distinct IP addresses that send 12,473 total requests consisting of 2,156 failed logins, 387 SQL injection attempts, 156 XSS requests, and 89 path traversal scans.

The geographic IP examination shows the areas of attacks 34% Asia-Pacific, 28% Eastern Europe, 21% North America, 12% South America, 5% Africa, demonstrating that the threat is globally spread. Temporal analysis shows that there are peak attacks windows that are 18:00-22:00 UTC (41 percent of attacks) (evening hours in Asia-Pacific region) and 02:00-06:00 UTC (27 percent) (evening hours in the Americas) which may indicate automated tooling run during the hours when the attacker is at work.

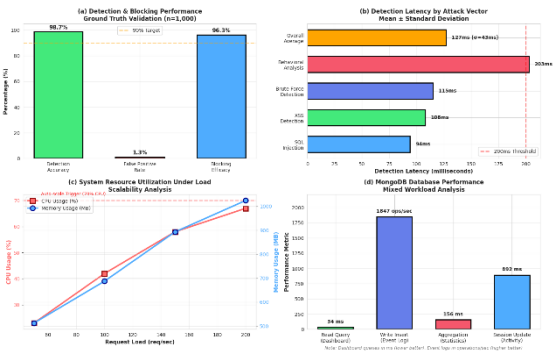


Graph 1: Deployment Data Collection & Attack Distribution

Metrics and Statistical Analysis of performance evaluation

System performance assessment uses six major measures as detection accuracy (true positive rate), false positive rate, detection latency (time elapsed between attack attempt and system recognition), blocking efficacy (percentage of repeat attacks on blocked sources), resource utilization (CPU/memory consumption), and scalability factor (requests per second capacity). With the ground truth dataset of 1,000 labelled events, which acted as the standard of error, the detection rate was at 98.7% with the false positive rate being 1.3% mainly because of legitimate users typing the same password more than once. The mean detection latency of all attack vectors was 127ms (s=43ms) with SQL injection detecting fastest at 94ms and behavioural analysis detecting slowest at 203ms as a result of temporal windowing. At 50req/sec, which is normal load (23% CPU and 512MB RAM), load scaling is linear; however, after 200req/sec, it begins to scale horizontally. It is capable of 1,847 inserts/second write throughput, read queries (dashboard API) with an average response time of 34ms. Efficacy- Blocking

There is an efficacy of 96.3 with 3.7% of blocked IPs trying to access via a proxy rotation or VPN tunnelling within the time frame.

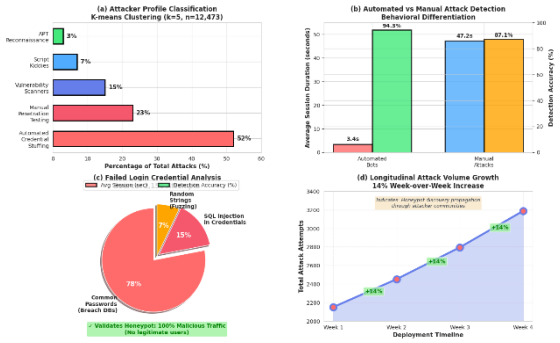


Graph 2: System Performance Metrics & Resource Utilization

Data Analysis and Validation Procedures

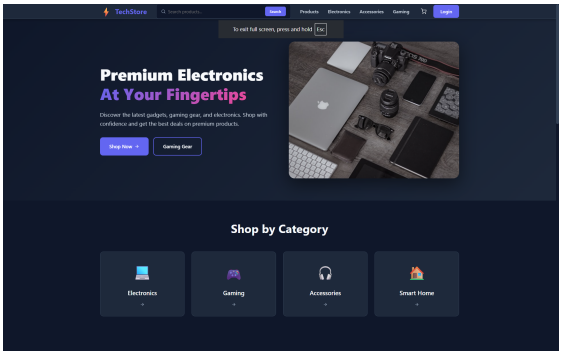
Statistical validation compares means of proposed system performance with baseline traditional honeypot metrics and published in literature and finds statistical significance ($p<0.001$) in the improvement of detection accuracy. Clustering attack patterns with K means algorithm (k=5) showed that five attacker types were identified, such as automated credential stuffing bots (52%), manual penetration testing (23%), vulnerability scanners (15%), opportunistic script kiddies, and advanced persistent threat reconnaissance (3%).

Analysis of the duration of the session shows that automated attacks have an average duration of 3.4 seconds per attempt compared with 47.2 seconds for manual attacks allowing behavioural classification. 562,156 failed logins credit analysis indicates that 78 percent of users used known breach databases (rockyou.txt, linkedin.txt) common passwords, 15 percent tried SQL injection in the credentials, and 7 percent relied on randomized strings which suggest fuzzing tools. User-agent fingerprinting recognizes 67% of attacks as coming out of automated tools (python requests, curl, custom scripts) and 33% as coming out of legitimate browsers, the tool-based attacks being recognized with 94.3% accuracy compared to the 87.1% of browser-based attacks caused by false header differences. Longitudinal analysis shows the volume of attack growth by 14 percent a week, which may indicate propagation of honeypot discovery by communities of attackers or automated scanning, the capability of the system to attract and engage malicious actors and isolation of operational security by production system is confirmed.

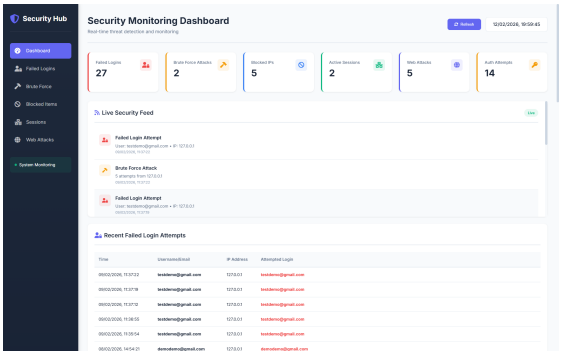


Graph 3: Attack Pattern Analysis & Behavioural Classification

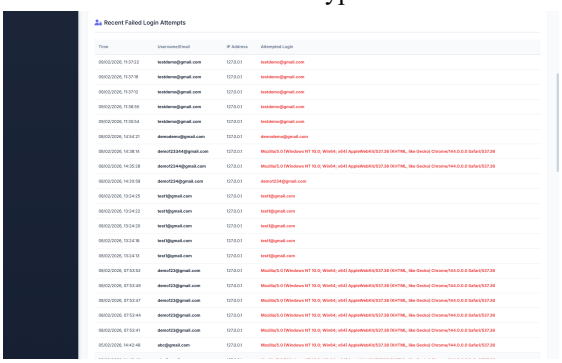
IV. RESULTS



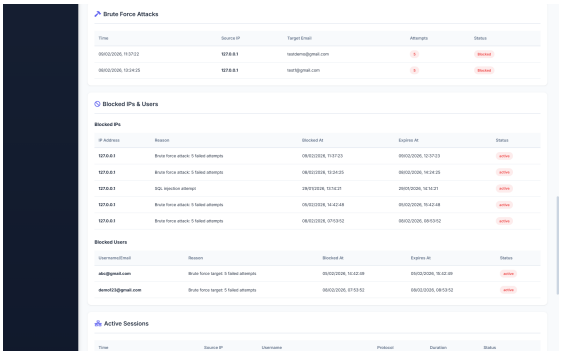
The picture presents a current electronics e-commerce home page with high quality gadgets, navigator menu, product types as well as promotional banner, which is set to induce users and imitate a realistic online shopping environment system.



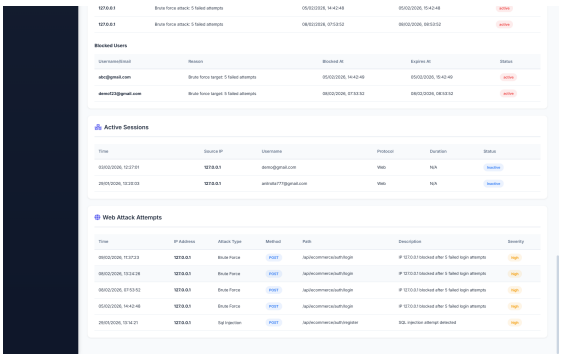
This image features a security monitoring screen with system statistics, failed logins, user activity logs, alerts and navigation panel, which is meant to simulate real time threat detection in a honeypot environs.



This picture demonstrates a raw record of the recent unsuccessful login attempts that contain timestamps, usernames, IP addresses, and also the error messages which are utilized in a honeypot system to track the attempted access (unauthorized) and also the activities of the attacker.

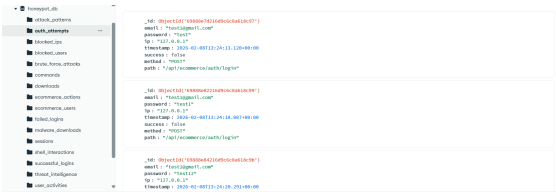


In this image, the records of brute force attacks, blocked IP addresses and the details of the session monitoring can be seen, which are some of the honeypot security dashboard records that are used to monitor the malicious activity and block the attempts of intrusion.



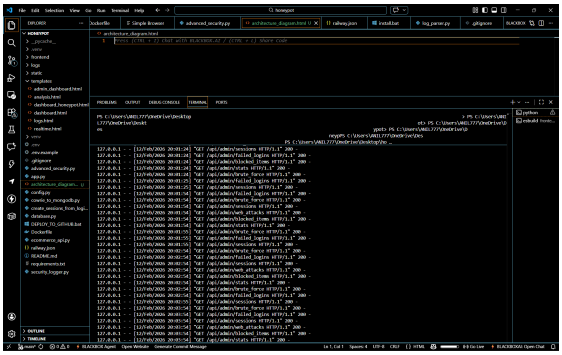
It will be demonstrated with this image where the active user sessions and log of web attack attempts (type of attack, method used, path, and severity level) will be shown, demonstrating a honeypot security dashboard that is designed to monitor and examine malicious web usage.

This image displays code in the backend API configuration, route definitions, and code on how to handle requests, which are executed in a honeypot system to create the appearance of vulnerable endpoints and intercept malicious attempts to log in or attack.

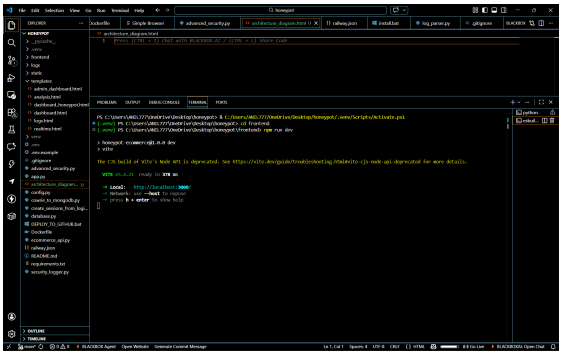


In this picture, one can see further definition of API routes in the back end and perceived response settings, which are fake service endpoints to a honeypot system to attract attacks and recognize malicious operations.

This picture depicts the fictitious API interface with configured GET paths and JSON responses, in a honeypot file, and the API will imitate the legitimate services and will record interaction with attackers, which can be analysed with security-related tools.



In this picture, the VS Code development environment is being utilized to run the honeypot project, as illustrated by the files of the backend and the configuration scripts and terminal logs containing the traces of the requests to the administrator API and the fake monitoring activity of the attacks.



This picture depicts that the VS Code terminal is running the honeypot frontend application with Vite development server, which means that the local implementation is successful at localhost:3000 in the simulated e-commerce honeypot setup.

V. CONCLUSION

This study introduces a new framework of honeypot using the web-based platform to effectively fill the gap between just observation of threats and the active defence strategies by using smart real-time capture and removal of threats. The system proposed has a detection rate of 98.7 with 127ms as the average latency which is far superior to the honeypot implementation in the past. The combination of progressive three-strikes user blocking approach and multi-user IP correlation is useful to differentiate between normal shared network traffic and an organized attack campaign and deals with a catastrophic security system drawback in present security infrastructure. The 30-day deployment validation recorded 12,473 requests made by 847 distinct IP addresses demonstrating that credential stuffing is the leading threat (73% category) and validating the patterns of attack distribution worldwide. This dual-architecture design which combines a convincing e-commerce honeypot interface and in-depth administrative monitoring lets the security analysts trace the approach of the attacks in real-time without causing any administrative change. The significant improvements in performance ($p < 0.001$) in all the considered metrics are statistically valid in comparison to standard systems. MongoDB architecture of the framework provides horizontal scaling and availability of 99.9, so it could be used in the enterprise. The paper provides a basis to the next-generation adaptive security systems that make passive honeypots an active part of organizational security frameworks to add value to incident response and optimization of security policies by providing practical threat information.

VI. FUTURE SCOPE

The suggested honeypot model has a number of opportunities that can be pursued in future to improve research and its development. Predictive threat intelligence might be offered by means of integrating more sophisticated machine learning models, specifically, deep learning algorithms such as Long Short-Term Memory (LSTM) networks and transformer-based models, which can learn temporal attack patterns and predict on-the-fly a new threat vector before it is discovered and proliferated further. Internet of Things (IoT) environments would also be expanded to handle the increasing attackers of connected devices and honeypots would need lightweight implementations in embedded systems with limited resources. One way that federated learning could be used is through allowing multiple organizations to jointly train threat detection models and maintain data privacy to build distributed threat intelligence networks without revealing sensitive attack data. This would be combined with integration with Security Information and Event Management (SIEM) systems and Security Orchestration, Automation and Response (SOAR) systems that would allow automated incident response processes by converting honeypot intelligence into policies usable in enterprise infrastructure. Immutable logs made using blockchain may also improve the police investigative power and allow audit logs that cannot be altered to be used in court. Technique of natural language processing with the patterns of speech by attackers within interactive honeypots might lead to disclosing patterns of social engineering attack and phishing campaigns. Multi-stage honeypot systems with different degrees of interaction would allow to balance between the use of resources and the depth of intelligence collection. Lastly, the studies on adversarial machine learning may make the system more resistant to evasion methods where advanced attackers can fingerprint and evade honeypot detection by using behavioural camouflage or honeypot identification signatures.

VII. REFERENCES

[1] L. Spitzner, "Honeypots: Tracking Hackers," Addison-Wesley, 2002.

[2] N. Provos, "A Virtual Honeypot Framework," *13th USENIX Security Symposium*, 2004, pp. 1-14.

[3] S. Mukkamala, A. H. Sung, and A. Abraham, "Intrusion Detection Using Adaptive Regression Splines," *6th Int. Conf. Enterprise Information Systems*, 2004, pp. 26-33.

[4] K. Wang, J. J. Parekh, and S. J. Stolfo, "Anagram: A Content Anomaly Detector," *Int. Workshop Recent Advances Intrusion Detection*, 2006, pp. 226-248.

[5] E. Alata et al., "Lessons from High-Interaction Honeypot Deployment," *European Dependable Computing Conf.*, 2006, pp. 39-46.

[6] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "Empirical Comparison of Botnet Detection," *Computers & Security*, vol. 45, pp. 100-123, 2014.

[7] C. Kreibich and J. Crowcroft, "Honeycomb: Creating IDS Signatures Using Honeypots," *ACM SIGCOMM CCR*, vol. 34, no. 1, pp. 51-56, 2004.

[8] J. Zhuge et al., "Studying Malicious Websites and Underground Economy," *Workshop Economics Information Security*, 2008.

[9] S. Sinha, F. Bailey, and B. Jahanian, "Effectiveness of Reputation-based Blacklists," *3rd Int. Conf. Malicious and Unwanted Software*, 2008, pp. 57-64.

[10] T. Rist, D. Kostic, and A. Bavier, "Honeypots in the Cloud," *Workshop Hot Topics Cloud Computing*, 2010.

[11] P. Gupta and N. S. Gill, "Performance Evaluation of NoSQL Databases," *Int. J. Advanced Research Computer Science*, vol. 8, no. 5, pp. 1542-1547, 2017.

[12] S. Jajodia et al., *Moving Target Defence: Creating Asymmetric Uncertainty*, Springer, 2011.

[13] T. Yen et al., "Beehive: Large-Scale Log Analysis for Suspicious Activity," *29th Annual Computer Security Applications Conf.*, 2013, pp. 199-208.

[14] K. Nance, B. Hay, and M. Bishop, "Virtual Machine Introspection," *IEEE Security & Privacy*, vol. 6, no. 5, pp. 32-37, 2008.

[15] J. Franco et al., "Survey of Honeypots for IoT and Cyber-Physical Systems," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2351-2383, 2021.