ISSN: 2582 - 6379
IJISEA Publications
International Journal for Interdisciplinary Sciences and Engineering Applications
IJISEA - An International Peer- Reviewed Journal
2025, Volume 6 Issue 2
www.ijisea.org

Secure Web Application Embed a Honeypot Link Using AWS

M.Sreedevi[1], A.Harini [2],CH.Yogesh Nithin[3], CH.Lakshmi Prasanna[4],CH.Chinni Raghava[5]
Assistant Professor[1],UGStudent[2,3,4,5]
Computer Science and Engineering
Amrita Sai Institute of Science & Technology
Paritala, Andhra Pradesh , India

ABSTRACT

Modern web applications face increasing threats from attacks such as SQL injection, XSS, and automated bots. This project enhances web security by integrating layered defense strategies including stealth honeypots, POST-based route authentication, and real-time attacker logging. A deceptive dashboard link and hidden URL routes silently trap and monitor attackers. Unauthorized access attempts to protected routes like the admin panel are redirected to a secure POST-authenticated page. Incorrect attempts trigger silent logging of attacker data including IP, email, timestamp, and user-agent. A secondary honeypot embedded in the authentication flow ensures continuous monitoring, while a concealed /admin_settings route responds passively yet logs intrusion attempts. This multi-pronged approach offers deeper threat detection, behavioral insights, and reinforces the resilience of modern web applications.

Key words:AWS, Python, Honeypot, WAF

I.INTRODUCTION

Web applications are critical assets for businesses and individuals, yet they face growing threats such as SQL injection (SQLi), cross-site scripting (XSS), and automated bot attacks. These attacks can compromise data, impact performance, and disrupt services. This project enhances web security by integrating two targeted honeypots: one concealed within the dashboard and another embedded in the URL structure. These honeypots silently detect and log unauthorized access attempts, enabling real-time threat analysis. Strict access control and request validation reduce false positives and ensure interaction only through valid user flows. This dual-honeypot approach strengthens early threat detection, supports informed incident response, and improves overall web application resilience.

II.RELATED WORK

Hoffman, A. (2020) – Web Application Security: Exploitation and Countermeasures for Modern Web Applications[1]

This book offers a detailed understanding of modern web application threats and how Web Application Firewalls (WAFs) play a central role in mitigating attacks such as SQL injection, cross-site scripting (XSS), and remote code execution. The author emphasizes how WAFs filter, monitor, and block malicious HTTP traffic based on rule sets, helping secure applications at the edge. From this work, I adopted the concept of leveraging WAFs as the first line of defense in cloud-deployed applications. My project builds upon this foundation by integrating AWS WAF into a Flask-based web application. However, I innovated by embedding honeypot logic within the application itself, which works in tandem with AWS WAF. When malicious behavior is detected through honeypot triggers, the application can dynamically flag suspicious IP addresses, which can later be used to refine WAF rules. This approach extends the concept of traditional static rule-based protection into a more dynamic, behavior-aware system.

Sullivan, B., & Liu, V. (2011) – Web Application Security: A Beginner's Guide[2]

This guide introduces foundational security principles for web developers and system architects, explaining secure coding, access control, and the integration of WAFs into the development lifecycle. A key takeaway from this book is the idea that application security must be incorporated from the design phase, not just as an afterthought. I applied this by building in multi-layered access control mechanisms, such as URL password locks and session validation, from the start of development. Where I innovated is in introducing voice-based access verification as an experimental, biometric-inspired mechanism. This goes beyond traditional password authentication by allowing only users who can verbally speak the passphrase "KannaAppa" to gain access to critical admin routes. Combining this with honeypot traps for incorrect access attempts represents a novel method of layered, deceptive, and adaptive access control.

Stuttard, D., & Pinto, M. (2011) – The Web Application Hacker's Handbook[3]

This book, often regarded as the bible for web penetration testers, deeply explores the techniques used by attackers to exploit security flaws in web applications. A significant portion is dedicated to understanding how attackers bypass WAFs and exploit logic flaws. The knowledge gained from this work influenced my strategy in designing honeypot endpoints disguised as legitimate admin functionality.for example, a hidden /admin_settings route that appears functional but is, in fact, a decoy trap. When unauthorized users access this page, their actions are silently logged along with their IP address, email (if logged in), and timestamp. Unlike traditional intrusion detection systems, this method provides attacker intelligence without alerting them, thus allowing ongoing behavior profiling. This application-level deception technique is a direct evolution of what the book teaches about attacker behavior, applied innovatively within the app architecture.

Collins, M. S. (2014) – Network Security Through Data Analysis[4]

Collins highlights how proactive data analysis can transform network security from reactive to predictive. Through real-world examples, the author shows how logs, alerts, and traffic data can be mined to identify patterns that indicate intrusion attempts. Inspired by this, my project includes a comprehensive attacker logging module that captures access attempts to honeypot routes, incorrect URL lock attempts, and failed voice authentication. This data is stored in a structured format and can be visualized through an admin dashboard, allowing administrators to track patterns, frequency, and sources of attempted breaches. While the original book focuses on network-level data, I have extended the concept to application-layer intelligence, giving web developers insight into attacker behavior in real time, with the potential to feed this information back into automated security responses.

III.METHODOLOGY

System Architecture

overview of AWS WAF and Honeypot integration

Deploy AWS WAF to Protect the Web Application: Implement AWS Web Application Firewall (WAF) to filter and monitor incoming HTTP/HTTPS requests, providing protection against common web threats such as SQL injection, cross-site scripting (XSS), and bot attacks.

Embed a Honeypot Link for Threat Analysis: Integrate a honeypot link within the web application to attract and monitor potential attackers. This will allow for the collection of valuable data on unauthorized access attempts, including IP addresses, attack methods, and patterns.

Enhance Threat Detection and Response: Utilize AWS WAF in conjunction with the honeypot to improve the detection and response to security incidents. The aim is to correlate honeypot interactions with WAF logs for a deeper understanding of attack behavior and to update security rules accordingly.

Improve Application Security Posture: Strengthen the overall security of the web application by leveraging the defensive capabilities of AWS WAF and the proactive data collection offered by the honeypot. The combination ensures a comprehensive and multilayered security approach.

Minimize False Positives and Optimize Performance: Fine-tune AWS WAF rules and honeypot configurations to reduce false positives, ensuring that security measures do not negatively impact user experience or application performance.

System Implementation

STEP-BY-STEP IMPLEMENTATION

1. CREATE A HOSTING ENVIRONMENT:

SET UP A WEB SERVER: Explain the process of setting up a web server (e.g., using AWS EC2 or another cloud provider).

CONFIGURE FOR STATIC CONTENT: Include details on how to configure the server to serve static content, such as HTML, CSS, and JavaScript files.

LOGGING AND MONITORING: Mention the importance of enabling logging (using tools like AWS CloudWatch) and monitoring the server for any anomalies or issues.

2. EMBEDDING A HONEYPOT LINK:

DESIGN THE HONEYPOT LINK: Choose a URL that attackers might target (e.g., /admin-access). Ensure that this link isn't accessible or visible to regular users.

IMPLEMENT THE HONEYPOT: Describe how to embed the link into the web application, in locations where attackers are likely to find it.

SAMPLE CODE: Provide a snippet of code that demonstrates how to create and embed the honeypot link within your HTML code.

3. TESTING AND FINAL DEPLOYMENT:

DEPLOY IN PRODUCTION: Discuss how to move the web application from a development environment to a production server, ensuring that all configurations (such as AWS WAF rules) are correctly applied.

CREATE A HOSTING ENVIRONMENT

1. SET UP A WEB SERVER:

CHOOSE A HOSTING PROVIDER: AWS EC2, DigitalOcean, or any cloud hosting platform can be used.

PROVISION THE SERVER: Create a virtual machine (VM) or an instance to host the web application. On AWS, this can be done using EC2 instances.

INSTALL WEB SERVER SOFTWARE: Install the necessary software such as Apache, Nginx, or any other web server to serve the web application.

2. CONFIGURE FOR STATIC CONTENT:

DIRECTORY SETUP: Organize directories for static content such as HTML, CSS, JavaScript, and image files.

ENABLE STATIC CONTENT DELIVERY: Configure the web server (e.g., Nginx or Apache) to serve static content from specified directories efficiently.

3. LOGGING AND MONITORING:

ENABLE SERVER LOGS: Enable access logs and error logs on the web server to track requests and issues.

INTEGRATE MONITORING TOOLS: Use AWS CloudWatch, Prometheus, or any monitoring tool to track server health, uptime, and performance metrics.

4. STEPS FOR CREATING A HOSTING ENVIRONMENT:

1.CHOOSE A HOSTING PROVIDER: Select a cloud provider (e.g., AWS, Digital Ocean,chey or Google Cloud) for your hosting needs.

PROVISION A VIRTUALMACHINE/INSTANCE: Spin up a virtual machine (VM) using the cloud provider's dashboard (e.g., AWS EC2) and choose the appropriate instance type (e.g., t2.micro for AWS).

SET UP WEB SERVER SOFTWARE:

Install a web server (e.g., Apache or Nginx).

Ensure that the web server is set up to start automatically when the VM starts.

CONFIGURE STATIC CONTENT DELIVERY:

Set up appropriate folders and permissions for storing static files like HTML, CSS, and JS.

Modify web server configurations to serve static content from the correct directory

ENABLE LOGGING AND MONITORING:

Enable access and error logging in the web server configuration.

Use a tool like AWS CloudWatch to gather log data and monitor the server's performance and health.

IV.RESULTS

EXCPETED OUTPUTS

SCREENSHOT 1: LOGIN AND REGISTER FORM WITH AWS IP ADDRESS

Display the login and registration form page of the website with the deployed IP address , where users can securely register and log in. Indicate that it's protected by AWS WAF and uses HTTPS for secure communication.

App window pop- up

SCREENSHOT 2: ATTACK THROUGH URL

Through URL also the hacker can't access the data from the admin panel . Incase of using this url way for accessing data that time also the unauthorization will be pop-up and also the attackers data will be recorded because we placed an second honeypot in the url.

user interface

Attendance screen

SCREENSHOT 3 : ATTACK THROUGH URL (POST Method Protection)

Through URL also the hacker can't access the data from the admin panel . Incase of using this url way for accessing data that time also the unauthorization will be pop-up and also the attackers data will be recorded because we placed an second honeypot in the url.

The system is designed to prevent unauthorized access even if an attacker attempts to directly reach the admin_panel route through the URL.

When a hacker manually enters the admin_panel route: No unauthorized access alert is shown immediately, maintaining a stealth approach. Instead, the attacker is redirected to a password authentication page, appearing as a normal security measure.

This authentication form operates via a POST method, ensuring that credentials are securely transmitted and preventing password exposure through the URL.

Access control behavior:

If the correct password is submitted via the POST method, the user is granted access to the Admin Dashboard.

ISSN: 2582 - 6379
IJISEA Publications
International Journal for Interdisciplinary Sciences and Engineering Applications
IJISEA - An International Peer- Reviewed Journal
2025, Volume 6 Issue 2
www.ijisea.org

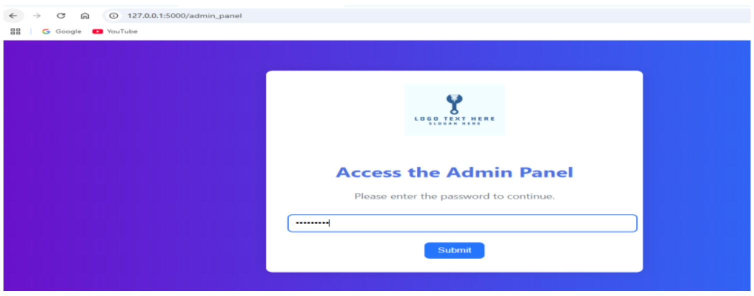

127.0.0.1:5000/admin_panel
Google
YouTube
LOGO TEXT HERE
Access the Admin Panel
Please enter the password to continue.
Submit

Figure 1:Shows Dashboard

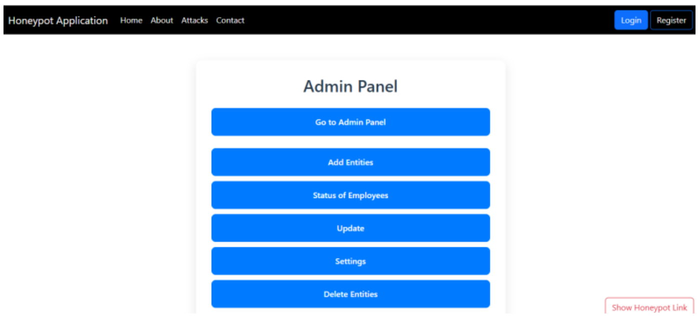


Figure 2:Shows Screenshot 4

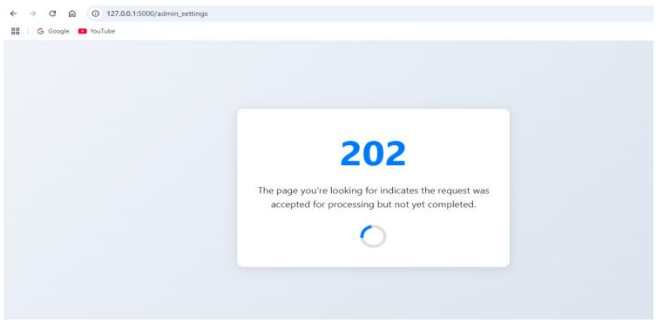
SCREENSHOT 4: ATTACK THROUGH URL

The /admin_settings route is designed as a silent honeypot.

When an unauthorized user attempts to access this route, the application does not trigger any visible alert or unauthorized access message to avoid raising suspicion.

Instead, the server responds with a 202 Accepted HTTP status code, indicating that the request was received and understood but not processed further.

Simultaneously, the system logs the access attempt in the database, capturing critical attacker details such as the user's email, IP address, timestam p, and user agent for further security analysis.



202

The page you're looking for indicates the request was accepted for processing but not yet completed.

Figure 3:Shows Screenshot 2

SCREENSHOT 2: ATTACKERS DATA

Show the attackers data page, which contains detailed logs about malicious attempts. Include information like login attempts through the honeypot, attack patterns, or vulnerability probes.

V.CONCLUSIONS

This project presents a robust and intelligent web application security framework by integrating AWS WAF with a dual-honeypot strategy. The honeypots—one embedded in the dashboard and another hidden in the URL structure—silently monitored and logged attacker behavior, aiding in proactive threat analysis. AWS WAF served as the primary defense, effectively mitigating threats like SQL injection, XSS, and bot attacks through customized, adaptive rules. Integration with AWS CloudWatch, CloudTrail, and Lambda enabled real-time monitoring, automated incident response, and forensic auditing. The system maintained high performance and user experience while offering scalable protection against evolving cyber threats. Overall, the project demonstrates how combining proactive honeypots with reactive WAF defenses creates a resilient and adaptive security model for modern web applications.

ACKNOWLEDGEMENT(S)

It gives me immense pleasure to thank my guide Professor of CSE Dr.M.Sreedevi, M Tech, (PhD), for having been a source of constant inspiration, precious guidance, and generous assistance during the project work. I deem it as a privilege to have worked under his able guidance. Without his close monitoring and valuable suggestions, this work would not have taken this shape. I feel that his help is unforgettable.

I take the privilege to record my thanks to H.O.D of CSE Dr. P. CHIRANJEEVI PhD for his encouragement, cooperation, and valuable suggestions to complete project in time.

I am very thankful to honorable Principal Dr M. SASIDHAR PhD and I express my sincere gratitude and deep sense of respect for making me available with all the required assistance and for his support and inspiration to carry out this project in the Institute.

I am thankful to the teaching and non-teaching staff of CSE department for their direct as well as indirect help in my project.

I am elated to avail myself to this opportunity to express my deep sense of gratitude to my parents

REFERENCES:

[1]Hoffman, A. (2020). Web Application Security: Exploitation and Countermeasures for Modern Web Applications. No Starch Press.

Description: This book covers various web application security topics, including an in-depth understanding of how Web Application Firewalls (WAFs) function and their role in mitigating common vulnerabilities like SQL injection, XSS, and other web-based attacks.

[2]Sullivan, B., & Liu, V. (2011). Web Application Security: A Beginner's Guide. McGraw-Hill Education.

Description: This guide offers a thorough introduction to securing web applications, including a section on WAFs, their implementation, and how they help protect applications from common attacks and advanced threats.

[3]Stuttard, D., & Pinto, M. (2011). The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws (2nd ed.). Wiley.

Description: Although this book focuses primarily on web application vulnerabilities and penetration testing, it also provides insights into how Web Application Firewalls can be circumvented and their role in mitigating attacks.

[4]Collins, M. S. (2014). Network Security Through Data Analysis: From Data to Action. O'Reilly Media.

Description: This book focuses on security analysis and data-driven decision-making in network and web security. It covers techniques that complement WAFs, including anomaly detection, which can be integrated into WAFs for enhanced security.

[5]Amazon Web Services (AWS). (2024). AWS WAF Developer Guide. Retrieved from https://docs.aws.amazon.com/waf

[6]Spitzner, L. (2003). Honeypots: Tracking Hackers. Addison-Wesley Professional.

[7]Krawetz, N. (2004). Introduction to Honeypots. Retrieved from https://www.securityfocus.com/infocus/1690

[8]Patel, A., Taghavi, M., Bakhtiyari, K., & Junior, J. C. (2013). An intrusion detection and prevention system in cloud computing: A systematic review. Journal of Network and Computer Applications, 36(1), 25–41.

[9]Schneier, B. (2015). Data and Goliath: The Hidden Battles to Collect Your Data and Control Your World. W.W. Norton & Company.

[10]Cloud Security Alliance (CSA). (2023). Cloud Security Guidance: Best Practices for Securing Applications. Retrieved from https://cloudsecurityalliance.org/