****

Technical Artefacts

**Sonam Pelden**

Table of Contents

[Preprocessing Pipeline 3](#_Toc187423318)

[Prediction Workflow 4](#_Toc187423319)

[AWS Deployment 5](#_Toc187423320)

[Virtual Private Cloud 5](#_Toc187423321)

[Elastic IP (EIP) 5](#_Toc187423322)

[Justification 7](#_Toc187423323)

[Challenges and Mitigation Strategies 9](#_Toc187423324)

[Conclusion 10](#_Toc187423325)

[References 11](#_Toc187423326)

### Preprocessing Pipeline



Figure 1 Pre-processing Pipeline

For machine learning model analysis, input text needs preparation (Kadhim 2019). This involves these methods:

1. **Converting Text to Lowercase**:
   * This stage standardises the text, ensuring that capitalisation variations have no effect on analysis.
   * Example: "Error in Input" becomes "error in input."
2. **Removing Special Characters and Numbers**:
   * Special characters and numbers have been eliminated to focus on essential words in the text.
   * Example: "Error#123" becomes "error."
3. **Stopword Removal**:
   * By eliminating "is," "and," and "the," NLTK improves readability.
   * Enhances data clarity and sentence focus.
4. **Lemmatization**:
   * The lemmatizer in NLTK simplifies words.
   * Example: "running" becomes "run."
   * This ensures word representation consistency.

With the preparation pipeline, input text is cleaned and formatted to fit the vectorizer utilised training.

### Prediction Workflow

A screenshot of a computer code

Description automatically generated

Figure 2 Prediction Workflow

Prediction workflow steps:

1. **Input Acceptance**:
   * A web form collects text data like logs and vulnerability descriptions.
2. **Preprocessing**:

* **Preprocess\_text** cleans and standardises input.

1. **Vectorization**:
   * Preprocessed text is converted to machine learning model-compatible numbers using vectorizer.transform. (Sharma *et al.* 2023)
   * The dataset-trained vectorizer (TF-IDF) is used to create models in this step.
2. **Prediction**:
   * The **Random Forest model** uses numerical data.
   * This model determines vulnerability type using input features.
3. **Decoding**:
   * Using a **label encoder**, numerical predictions are converted to readable format.
   * Example: A numeric label "1" is decoded as "SQL Injection."
4. **Dynamic Result Display**:
   * To give users immediate feedback, the admin page dynamically displays the decoded result.

**Justification of Flask and Python**

* **Flask**:
  + Flask is lightweight and versatile, making it ideal for tiny to medium-sized projects like this page (Pegado, Rocha & Barata, 2024).
  + Easy integration with Jinja2 for dynamic HTML rendering and interactive user interfaces.
  + Its simplicity lets developers focus on developing functionality without excessive complexity.
* **Python**:
  + Python's NumPy, Pandas, and Scikit-learn environment makes data processing and machine learning efficient (Raschka, Patterson & Nolet, 2020).
  + Backend and machine learning are seamlessly integrated with Flask and Python.
  + Python's readability and community support make it ideal for rapid development and scalability.

# AWS Deployment

## Virtual Private Cloud

A **Virtual Private Cloud (VPC)** is a secure, isolated part of a public cloud that enables users to construct and manage resources in a virtualised network, such as virtual machines, databases, and applications (Yang *et al.* 2018). VPCs let enterprises control IP address ranges, subnets, routing tables, and security policies to optimise workload performance and security.

A key VPC feature is isolation. A VPC isolates cloud resources from other networks to protect data and traffic. Users can create private and public subnets for different applications and control internet access with NAT gates, internet gateways, or private network configurations for corporate reasons.

VPCs use security groups and NACLs as virtual firewalls. These elements improve data security and compliance (Park, Lee & Park, 2022). VPCs work well with load balancers, databases, and storage systems, making them ideal for hybrid clouds. They offer efficient and secure hybrid deployments by connecting on-premises infrastructure to the cloud via VPN or Direct Connect.

Finally, a Virtual Private Cloud delivers a secure, private network within the public cloud's scalability and cost-efficiency (Krishna, Srinivas & Reddy, 2023b). Virtual private clouds (VPCs) provide complete control over network settings and robust security for applications ranging from web hosting to complex enterprise workloads.

## Elastic IP (EIP)

A **Virtual Private Cloud (VPC)** is normally free, however some components may cost more dependent on usage (Nowakowski *et al.* 2017). Example: Elastic IP. Static IPv4 elastic IPs are for dynamic cloud computing. Users can maintain a constant public IP address even if underlying resources are stopped or resumed, such as EC2 instances. Applications can be accessed easily without altering IP configurations.

Elastic IPs are charged if they are not assigned to a running instance or if many are allocated (Harlap *et al.* 2017). Elastic IPs without any cost to prevent overallocation of limited IP resources. Elastic IPs enable low-cost, dependable connectivity when regulated and tied to running instances.

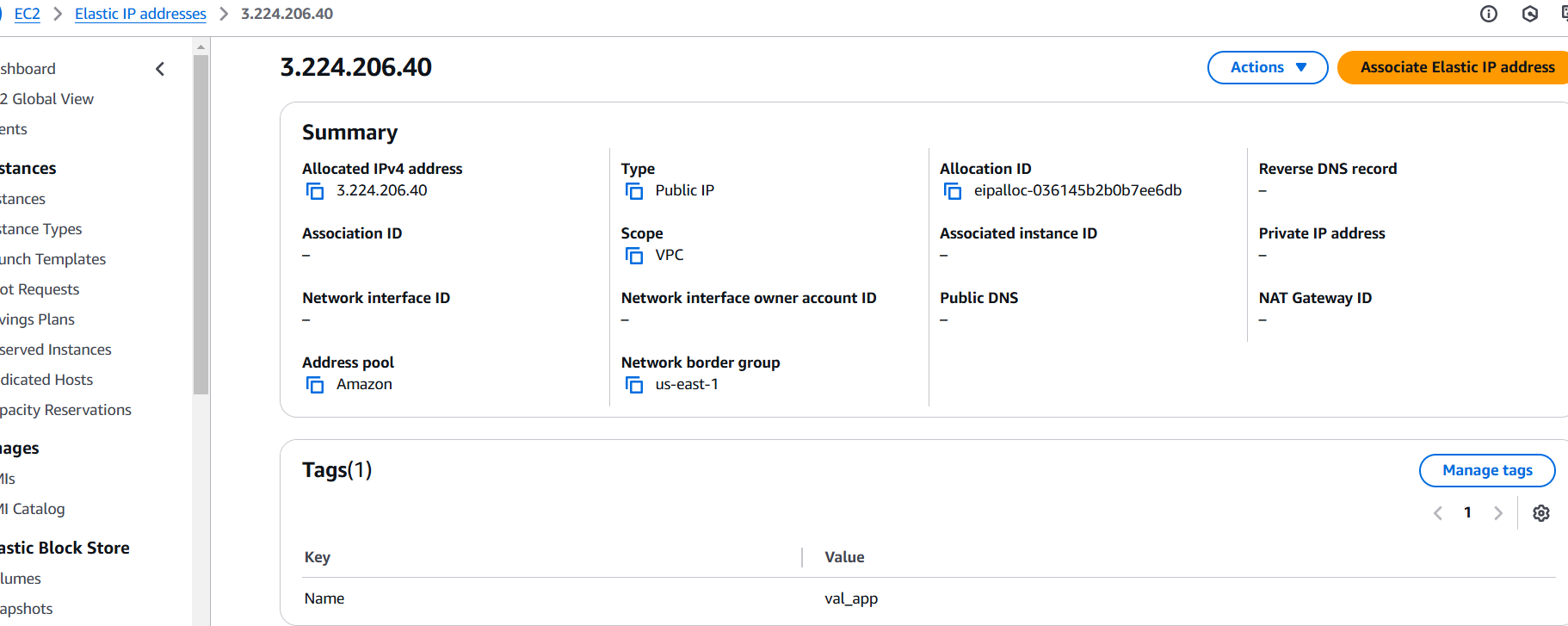


Figure 3 Virtual Private Cloud (VPC)

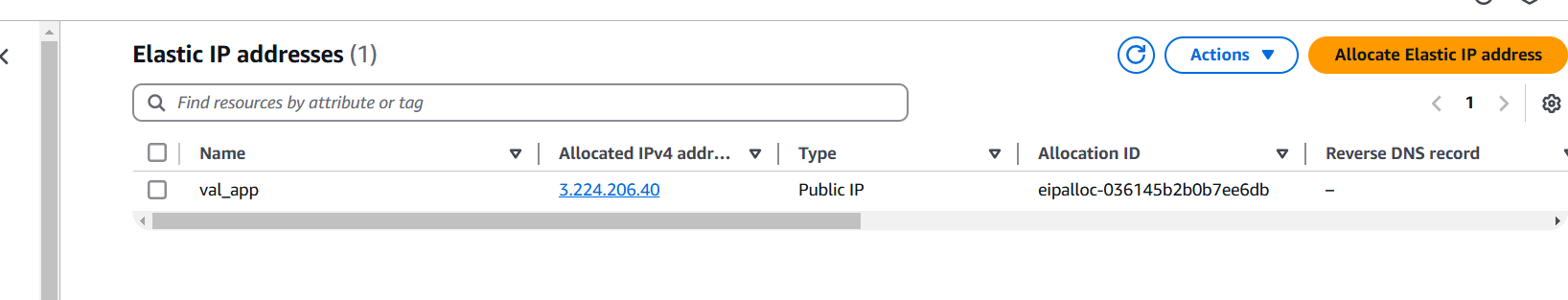


Figure 4 Elastic IP addresses

Val\_app, an assigned EIP, appears in the AWS EIP administration interface. AWS provides public IPv4 elastic IPs for EC2 instances to keep a fixed IP address. This functionality keeps connectivity even when the underlying instance is stopped, restarted, or replaced, making it helpful for stable external access applications.

The interface shows EIP details such Allocated IPv4 Address (3.224.206.40), Type (Public IP), and Allocation ID (eipalloc-036145b2b0b7ee6db). The EIP has no reverse DNS, indicating no resource. Use the "Actions" option to assign this EIP to an EC2 instance or other AWS resources for simple access to hosted apps.

Static DNS mapping and fixed IP applications require elastic IPs. Elastic IPs are limited by area, and AWS may punish EIPs that are unrelated to running instances or share resources. This billing method maximises resource consumption and discourages overuse.

Select Allocate Elastic IP Address in the interface to generate more. This with the ability to dynamically attach or detach EIPs from resources give network setup flexibility and control. Elastic IPs ensure reliable external access to AWS applications and scalable networks (Rauf et al. 2020).

# Justification

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component** | **Selected Method** | **Reason for Selection** | **Alternative Methods** | **Reason for Not Choosing Alternatives** |
| **Backend Framework** | Flask | - Lightweight and modular framework. - Easy to integrate with Python libraries and machine learning models. - Jinja2 supports dynamic HTML rendering. | Django | - More complex and heavyweight, providing unnecessary features for the small to medium-scale project. - Higher learning curve. |
| **Programming Language** | Python | - Extensive libraries (e.g., NumPy, Pandas, Scikit-learn) for machine learning and data processing. - Readability and large community support. | Java, R | - Java lacks Python's versatility in machine learning. - R is more suited for statistical analysis than web development. |
| **Machine Learning Model** | Random Forest | - High accuracy and interpretability. - Handles small to medium datasets effectively. - Less prone to overfitting compared to Neural Networks. | Neural Networks, SVM | - Neural Networks are resource-intensive and complex. - SVM underperforms for multi-class text classification tasks. |
| **Preprocessing Pipeline** | NLTK | - Efficient tools for stopword removal, lemmatization, and text cleaning. - Simple to implement and widely adopted in NLP projects. | SpaCy | - SpaCy is more advanced but adds unnecessary complexity for this project’s scope. |
| **Cloud Networking** | Virtual Private Cloud (VPC) | - Provides secure, isolated network environments. - Enables custom configurations for subnets, routing, and access controls. - Ideal for hybrid setups. | Public Cloud Configurations | - Lack the isolation and control needed for secure applications. - Unsuitable for hosting sensitive workloads. |
| **IP Address Management** | Elastic IP (EIP) | - Ensures consistent public IP access, even during resource restarts or reallocation. - Ideal for DNS mapping and stable connectivity. | Dynamic IPs | - Dynamic IPs do not provide stability for public-facing applications, complicating DNS configurations. |
| **Preprocessing Methods** | Stopword removal, lemmatization (NLTK) | - Simplifies text and ensures uniform representation. - Enhances compatibility with vectorizers. - Widely supported and efficient for small datasets. | No Preprocessing | - Without preprocessing, raw data would degrade model performance due to noise and inconsistencies. |
| **Deployment Platform** | AWS (VPC, S3) | - Scalable, reliable, and secure cloud infrastructure. - Integration with backup services for disaster recovery. | Google Cloud, Azure | - AWS offers better ecosystem integration with Python and Flask. - Team familiarity with AWS tools ensured smoother deployment. |

# Challenges and Mitigation Strategies

|  |  |
| --- | --- |
| **Challenge** | **Mitigation Strategy** |
| **Text Preprocessing Issues** | Used NLTK for lemmatisation and stopword elimination. Standardised input formats by removing special characters and digits for model and vectorizer compatibility. |
| **Model Integration and Compatibility** | Addressed compatibility issues by preloading pre-trained components (e.g., Random Forest model, TF-IDF vectorizer, and label encoder) using joblib. Conducted end-to-end testing to ensure seamless integration with Flask. |
| **Dynamic Input Handling** | Implemented robust input validation and error handling mechanisms to preprocess user inputs dynamically, ensuring consistency and reducing errors during prediction workflows. |
| **High Computational Requirements of the Random Forest Model** | Pre-optimized Random Forest model settings during training to save resources. To handle real-time forecasts without delays, optimised AWS deployment resources. |
| **Scaling and Resource Management in AWS VPC** | Configure VPC with security groups and subnets to isolate resources and optimise performance. Elastic IPs were associated with active instances to avoid wasteful expenses. |
| **Elastic IP Resource Underutilization** | Ensured Elastic IPs were always associated with running instances to minimize costs. Configured monitoring and periodic checks to prevent resource misuse. |
| **Dynamic Result Display Challenges** | Designed a front-end workflow where prediction results are dynamically displayed using Flask and Jinja2 templates. Regularly tested the admin page for responsiveness and ensured accurate real-time result updates. |
| **Ensuring Data Security in AWS Deployment** | Configured IAM roles, security groups, and network ACLs for controlled access. Enabled server-side encryption for Elastic IPs and VPC configurations to enhance data security and comply with best practices. |
| **Complexity in Preprocessing Pipeline Design** | Simplified the preprocessing pipeline by selecting essential text cleaning steps (lowercasing, lemmatization, and stopword removal). This ensured efficient processing without degrading model performance. |
| **Maintaining Cost-Efficiency for Cloud Deployment** | Used AWS pay-as-you-go pricing models to ensure cost-effective deployment. Periodically reviewed resource utilization and employed Elastic IP optimizations to minimize unnecessary expenditures. |

Top of Form

Bottom of Form

# Conclusion

Flask-based web application with machine learning and AWS services successfully predicted web vulnerabilities and managed cloud resources. Text data is cleaned and transformed into a machine learning model-friendly format via the Python and NLTK preprocessing pipeline. The Random Forest model was chosen for its durability and vulnerability type prediction accuracy. Flask was chosen as the backend framework because it is lightweight and modular, ensuring seamless Python library integration and speedy machine learning pipeline deployment.

Secure, scalable, and reliable access to cloud resources was made possible using **VPC** and **Elastic IP**. Elastic IP ensures application access during resource reallocation or restarts, while VPC provides a secure, isolated network. Project priorities include scalability, dependability, and cost-efficiency. Advanced machine learning algorithms and cloud resource management may improve performance in future iterations.

# References

1. B. Rauf, H. Abbas, A. M. Sheri, W. Iqbal and A. W. Khan 2020, 'Enterprise Integration Patterns in SDN: A reliable, Fault-Tolerant communication framework,' *IEEE Internet of Things Journal*,  vol. 8, pp. 6359–6371. DOI: 10.1109/jiot.2020.3034350.
2. D. Yang, H. Wei, Y. Zhu, P. Li & J. -C. Tan 2018, 'Virtual Private Cloud Based Power-Dispatching Automation System—Architecture and Application,' *IEEE Transactions on Industrial Informatics*, pp. 1756–1766. DOI: 10.1109/tii.2018.2849005.
3. Harlap, A., Tumanov, A., Chung, A., Ganger, G.R. and Gibbons, P.B., 2017, ‘**Proteus: agile ML elasticity through tiered reliability in dynamic resource markets**,' *Proceedings of the Twelfth European Conference on Computer Systems*, pp. 589–604. DOI: 10.1145/3064176.3064182.
4. Kadhim, A.I. 2019, 'Survey on supervised machine learning techniques for automatic text classification,' *Artificial Intelligence Review*, pp. 273–292. DOI: 10.1007/s10462-018-09677-1.
5. Krishna, D.S., Srinivas, G. and Reddy, P.V.G.D.P. 2023, 'Novel private cloud architecture: A three tier approach to deploy private cloud using virtual machine manager,' *Intelligent Decision Technologies*, pp. 275–285. DOI: 10.3233/idt-229035.
6. Nowakowski, P., Bubak, M., Bartyński, T., Gubała, T., Harężlak, D., Kasztelnik, M., Malawski, M. and Meizner, J., 2017, 'Cloud computing infrastructure for the VPH community,' *Journal of Computational Science*, pp. 169–179. DOI: 10.1016/j.jocs.2017.06.012.
7. Park, S.-J., Lee, Y.-J. & Park, W.-H. 2022, 'Configuration method of AWS security architecture that is applicable to the cloud lifecycle for sustainable social network,' *Security and Communication Networks*, 2022, pp. 1–12. DOI: 10.1155/2022/3686423.
8. Pegado, A.M., Rocha, A.D. & Barata, J. (2024) 'Enhancing software distribution and collaboration in manufacturing: developing the IMOS platform,' *IEEE Access*, vol. 13, pp. 480-494. DOI: 10.1109/access.2024.3521867.
9. Raschka, S., Patterson, J. & Nolet, C. (2020) 'Machine learning in Python: main developments and technology trends in data science, machine learning, and artificial intelligence,' *Information*, p. 193. DOI: 10.3390/info11040193.
10. Sharma, B., Sharma, L., Lal, C. & Roy, S.*.* (2023) 'Explainable artificial intelligence for intrusion detection in IoT networks: A deep learning based approach,' *Expert Systems With Applications*, p. 121751. DOI: 10.1016/j.eswa.2023.121751.