**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

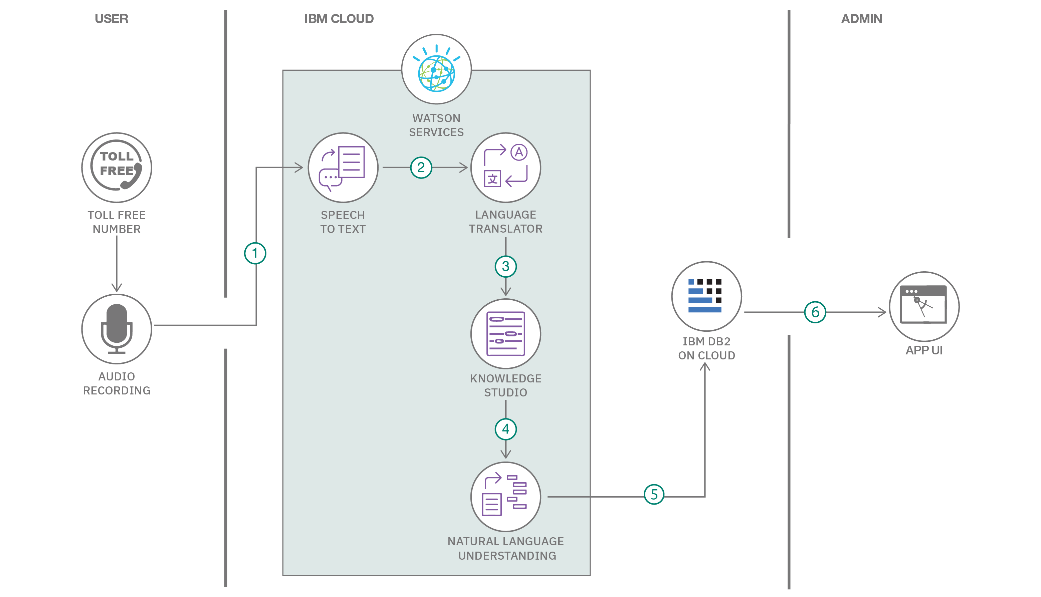
|  |  |
| --- | --- |
| Date | 25 October 2023 |
| Team ID | 1.1 |
| Project Name | AI-Enhanced Intrusion Detection System |
| Maximum Marks | 4 Marks |

**Technical Architecture:**

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

**Example: Order processing during pandemics for offline mode**

**Reference:** [**https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/**](https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/)

Guidelines:

1. Include all the processes (As an application logic / Technology Block)
2. Provide infrastructural demarcation (Local / Cloud)
3. Indicate external interfaces (third party API’s etc.)
4. Indicate Data Storage components / services
5. Indicate interface to machine learning models (if applicable)

**Table-1 : Components & Technologies:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
| 1. | Data Sources | Data sources can include network traffic data, system logs, and security event data from various devices. They provide the raw input for intrusion detection. | Network taps, sensors, log collectors, firewall logs, IDS/IPS devices, packet capture tools. |
| 2. | Data Processing | This component is responsible for ingesting, preprocessing, and normalizing the data to make it suitable for analysis. | Apache Kafka, Logstash, Fluentd, data normalization scripts. |
| 3. | Data Storage | A central repository for storing pre-processed data, enabling historical analysis and real-time access. | Elasticsearch, Hadoop HDFS, AWS S3, SQL/NoSQL databases. |
| 4. | Machine Learning Model | Machine learning models are used to identify anomalies and potential intrusions by learning from historical data. | Python (Scikit-Learn, TensorFlow, PyTorch), Jupyter Notebooks, GPU for deep learning. |
| 5. | Intrusion Detection Rules Engine | This component includes predefined rules and signatures for known attack patterns, aiding in signature-based detection. | Snort, Suricata, YARA rules, custom signatures. |
| 6. | Real-Time Analysis | This is where machine learning models and intrusion detection rules are applied to the data for real-time analysis. | Real-time stream processing frameworks (e.g., Apache Flink, Apache Kafka Streams), Python for rule execution. |
| 7. | Alert Management | Alerts generated by the real-time analysis component are managed here, prioritized, and routed for further investigation. | SIEM systems, alert correlation tools, ticketing systems. |
| 8. | User Interface (UI) | Provides a dashboard for security administrators and analysts to monitor the system's performance and respond to incidents. | Web-based UI (HTML, CSS, JavaScript), dashboard frameworks (e.g., Kibana, Grafana). |
| 9. | Reporting and Analytics | Generates reports and provides analytics on detected incidents, trends, and system performance. | Reporting tools (e.g., Tableau, Power BI), data analytics platforms (e.g., Apache Spark), custom analytics scripts. |
| 10. | Notification and Response | This component can automatically trigger predefined responses when intrusions are detected, such as isolating affected systems or blocking malicious traffic. | Automation scripts, APIs for response actions, orchestration tools (e.g., Ansible). |
| 11. | Integration APIs | APIs are used to integrate the intrusion detection system with other security tools and incident response platforms. | RESTful APIs, webhook integrations, third-party security APIs. |
| 12. | Scalability and Load Balancing | Ensures the system can handle high volumes of data and provides fault tolerance. | Load balancers, container orchestration (e.g., Kubernetes), cloud auto-scaling. |
| 13. | Security and Compliance | Various security measures and compliance considerations should be integrated throughout the architecture to ensure data and system security. | Encryption (TLS/SSL), access control, auditing and monitoring tools, compliance frameworks (e.g., PCI DSS, HIPAA). |

**Table-2: Application Characteristics:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Characteristics** | **Description** | **Technology** |
| 1. | Anomaly Detection | AI-enhanced IDS can detect anomalies in network traffic and system behaviour by learning what is considered normal. When it identifies deviations from this baseline, it can trigger alerts or automatic responses | Machine learning algorithms, such as neural networks, clustering, and statistical analysis. |
| 2. | Signature-Based Detection | The system is equipped with predefined signatures or patterns of known threats. When network traffic matches these signatures, it raises alarms or takes predefined actions. | Pattern recognition, rule-based systems. |
| 3. | Real-time Monitoring | AI-enhanced IDS systems continuously monitor network traffic in real-time, providing immediate threat detection and response capabilities. | Continuous packet analysis and traffic monitoring. |
| 4. | Behaviour Analysis | It analyses user and system behaviour to identify suspicious activities or deviations from normal behaviour patterns. | Machine learning, data analysis. |
| 5. | Regular Updates and Training | The IDS continuously updates its knowledge base and machine learning models to adapt to evolving threats. | Data collection and model retraining. |
| 6. | User-Friendly Interface | A user-friendly interface allows security analysts to interact with and configure the IDS efficiently, helping them make informed decisions. | User experience (UX) design principles. |
| 7. | Threat Intelligence Integration | The IDS can stay updated with the latest threat intelligence, incorporating new threat signatures and indicators of compromise into its detection capabilities. | API integrations with threat feeds and databases. |
| 8. | Scalability | The system can be scaled horizontally and vertically to accommodate the needs of various network sizes and configurations. | Distributed architecture, cloud-based solutions. |

**References:**