



Data Science and Analytics

ORG Data Thread and Weakness Analysis

Student 1: Vijay Singh (7025700)
Student 2: Manoj S (7025700)
Student 3: Vatsal Mahajan (7025700)
Course of Studies: Industrial Informatics

First examiner: Prof. Dr. Elmar Wings
Second examiner: Prof. Dr. Walter Colombo

Submission date: January 5, 2025

Contents

Acronyms	v
1 Introduction	1
1.1 Case:	1
1.2 Challenges and Results:	1
2 Positioning Your Analytics	2
2.1 Existing Solutions:	2
2.2 State-of-the-Art:	2
2.2.1 Overview of Current Solutions:	2
2.2.2 Capabilities and Limitations:	2
2.2.3 Relevance to Analytics:	2
2.2.4 Emerging Trends:	3
3 Application Sector	4
3.1 Application Sector:	4
4 Purpose of Your Analytics	5
4.1 What It Does:	5
5 Positioning Within Standard Reference Architectures	6
6 Connected Infrastructure Components	8
6.1 Infrastructure Integration:	8
7 Major Requirements	9
7.1 Requirements:	9
8 Knowledge Discovery in Databases (KDD) Process	10
8.1 Topic Description	10
8.1.1 Special Challenges	10
8.2 Database	11
8.3 Data Selection	11
8.3.1 Origin	11
8.3.2 Data Format	11
8.3.3 Features	12
8.3.4 Size	12

Contents

8.4	Data Preparation	12
8.4.1	Merging and Conversion	12
8.4.2	Initial Cleaning	12
8.4.3	Key Attributes	13
8.5	Data Transformation	13
8.6	Data Mining	14
8.6.1	Feature Extraction	14
8.6.2	Clustering	14
8.6.3	Anormalities	15
8.7	Model	15
8.7.1	DBSCAN	15
8.7.2	Isolation Forest	15
8.8	Validation/Verification	16
8.8.1	DBSCAN Validation	16
8.8.2	Isolation Forest Validation	16
8.9	Data Visualization	16
8.10	Conclusion	16
	Bibliography	17

List of Figures

1.1	Original Data format	1
5.1	Mapping Functionalities in RAMI 4.0 and IIRA	7
8.1	KDD Process	11
8.2	Converted CSV file	13
8.3	Error log	13
8.4	Result after applying data transformation	14

List of Tables

Acronyms

1 Introduction

1.1 Case:

The focus of this analysis is to identify and mitigate potential threats and weaknesses within Orgadata's SimplyTag system. SimplyTag facilitates quick access to construction-related data through a web app, making it essential to safeguard against malicious actors attempting to exploit the system. This involves analyzing logs to detect suspicious activity and ensure data integrity.

1.2 Challenges and Results:

Challenges:

- Handling large-scale logs to pinpoint anomalies as shown in the figure 1.1.
- Differentiating legitimate user activity from malicious attempts.
- Establishing efficient protocols to respond to detected threats.

Results Achieved:

- Implementation of enhanced monitoring protocols using trace IDs and HTTP status codes.
- Reduced data breaches by identifying and blocking unauthorized requests.

[illegible]

Figure 1.1: Original Data format

2 Positioning Your Analytics

2.1 Existing Solutions:

The analytics developed for Orgadata's SimplyTag system are positioned uniquely when compared to conventional tools. Traditional solutions like **Intrusion Detection Systems** (IDS) and log monitoring platforms (e.g., Splunk, ELK Stack) offer general frameworks for detecting anomalies and breaches. However, they often lack customization tailored to specific applications. Orgadata's approach is distinguished by its precise utilization of trace IDs, HTTP status codes, user agent patterns, and paths to monitor requests in real-time.

2.2 State-of-the-Art:

2.2.1 Overview of Current Solutions:

- Existing tools such as Splunk, ELK Stack, and Intrusion Detection Systems (IDS) provide robust frameworks for monitoring and analyzing security events. These solutions excel at processing vast amounts of log data and identifying potential threats in general scenarios.
- Security platforms often use rule-based or heuristic approaches for anomaly detection but may struggle with domain-specific customizations.

2.2.2 Capabilities and Limitations:

- **Capabilities:** Tools like Splunk and ELK Stack provide scalability, integration options, and comprehensive dashboards for security analytics. IDS focuses on real-time threat detection.
- **Limitations:** Lack of precise tailoring for SimplyTag's requirements, such as analyzing specific HTTP status codes and user agent patterns. High false-positive rates and inability to integrate seamlessly with Orgadata's infrastructure are additional challenges.

2.2.3 Relevance to Analytics:

The SimplyTag analytics focus on filling these gaps by leveraging domain-specific insights, such as monitoring trace IDs and HTTP response patterns to identify

anomalies and unauthorized activity.

1. The use of trace IDs enables seamless tracking of individual requests across logs, allowing for detailed insights into potential vulnerabilities.
2. By monitoring HTTP status codes, the system identifies and flags suspicious patterns, such as unexpected 200 codes for unauthorized paths.
3. Integration of user agent analysis ensures that illegitimate devices or configurations can be quickly identified and addressed.

2.2.4 Emerging Trends:

- AI-driven anomaly detection is gaining traction, enabling systems to learn and adapt to evolving threats.
- Zero-trust architecture and real-time monitoring advancements align with SimplyTag's goals of enhancing data security and operational reliability.

This focused methodology bridges gaps left by traditional systems, providing a solution that is both targeted and scalable for SimplyTag's operational needs.

3 Application Sector

3.1 Application Sector:

This project belongs to the Industry domain, specifically the Construction Software sector, with a focus on::

- **Threat Detection:** Monitoring system logs to identify malicious activity and unauthorized data access.
- **Data Integrity Assurance:** Safeguarding construction-related data from breaches to ensure accurate and reliable information.
- **Operational Security Optimization:** Enabling proactive measures to address potential weaknesses and improve system resilience.

By contributing to the broader domain of secure digital construction tools, the project aligns with emerging trends like data-driven operational efficiency and advanced cybersecurity measures tailored for industry-specific applications.

4 Purpose of Your Analytics

4.1 What It Does:

This project belongs to the Industry domain, specifically the Construction Software sector, with a focus on:

- **Condition Monitoring:** Tracks HTTP requests in real-time to identify anomalies. This includes analyzing user agents and status codes to pinpoint irregularities that could indicate potential breaches or vulnerabilities.
- **Prevention:** Identifies and blocks malicious requests before they lead to breaches. Proactive security measures ensure that sensitive data and system integrity remain uncompromised.
- **Diagnosis:** Conducts post-incident analysis to refine future detection capabilities. By learning from past incidents, the analytics evolve to handle emerging threats more effectively.
- **Optimization:** Enhances system performance by ensuring secure operations without disrupting user experience. This involves balancing security measures with system usability, especially in high-demand environments like construction software solutions.

5 Positioning Within Standard Reference Architectures

This solution aligns with several modern frameworks and architectures:

- **Smart-City:** The analytics align with secure data handling standards in urban development contexts, ensuring efficient and secure information exchange in large-scale urban projects..
- **IIoT/IIRA:** The approach supports reliability and security within industrial systems, adhering to frameworks like the Industrial Internet Reference Architecture (IIRA) for seamless integration and operational consistency, refer figure 5.1.
- **RAMI 4.0:** Within the Reference Architecture Model for Industry 4.0 (RAMI 4.0), the analytics are positioned in the "**Information Layer**", where data processing, analysis, and anomaly detection occur. It also contributes to the "Functional Layer" by enabling actionable insights and decision-making based on real-time monitoring, refer figure 5.1.
- **ISO Standards Compliance:** The analytics are developed in alignment with international standards such as ISO 27001, ensuring robust security practices and data protection.
- **Cybersecurity Frameworks:** Incorporating principles from the NIST Cybersecurity Framework, the analytics bolster detection, prevention, and response mechanisms for threats.

These alignments ensure that SimplyTag analytics not only address Orgadata's specific needs but also conform to global benchmarks, making them scalable and applicable across various industrial and urban contexts.

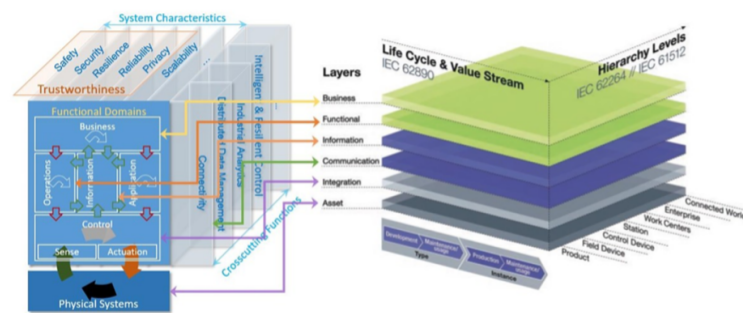


Figure 5.1: Mapping Functionalities in RAMI 4.0 and IIRA

6 Connected Infrastructure Components

6.1 Infrastructure Integration:

- **SimplyTag API Backend:** Serves as the primary source for log data. It collects and provides detailed information about HTTP requests, including trace IDs and status codes, for analysis.
- **Orgadata's Logikal System:** Acts as the operational backbone for data-driven decision-making. This system contextualizes log data by linking it with construction-related project details, enabling precise threat detection.
- **External Threat Detection Tools:** To enhance the analytics, integration with external tools such as SIEM platforms or advanced machine learning models for threat prediction can be implemented. These tools offer additional layers of analysis and improve overall system resilience.
- **Cloud Infrastructure:** SimplyTag leverages cloud infrastructure to ensure scalability and reliability. The cloud platform supports large-scale log storage and computational power required for real-time analytics.
- **Mobile and Web Interfaces:** The analytics extend to user-facing interfaces like the SimplyTag mobile and web apps, ensuring seamless user interaction while maintaining robust security measures.

By connecting these components, the analytics create a comprehensive security ecosystem that is robust, scalable, and aligned with Orgadata's operational goals.

7 Major Requirements

7.1 Requirements:

- **Structural:** The analytics must seamlessly integrate with existing SimplyTag and Orgadata infrastructure.
- **Behavioral:** Real-time anomaly detection with high accuracy and minimal false positives.
- **Functional:** Efficient parsing of HTTP status codes and trace IDs to identify suspicious activity.
- **Technological:** Implementation of advanced algorithms for log analysis and anomaly detection.

This structured approach ensures that the Threat and Weakness Analysis for Orgadata's SimplyTag system is robust, effective, and well-positioned within industry standards.

8 Knowledge Discovery in Databases (KDD) Process

KDD plays a pivotal role in helping organizations navigate the over whelming volumes of data generated in today’s information-driven world. KDD is a structural approach to data analysis to discover and make explicit knowledge available in extensive data sets [Win24]. This methodology involves a series of well-defined steps, including data selection, preprocessing, transformation and analysis, leading to the generation of actionable knowledge as shown in the figure 8.1. [MFH00]

KDD is particularly relevant for analyzing log file data in the context of threat and weakness analysis. The structured approach ensures that patterns and anomalies are identified efficiently, enabling the identification of targeted solutions for improving system security and operational performance. [FPS97]

In this project, the KDD process will serve as the backbone for organizing and implementing analytics tasks, ensuring a systematic exploration of the data to uncover valuable insights.

8.1 Topic Description

This section of the report outlines the objective to focuses on identifying anomalies, vulnerabilities, and inefficiencies within the dataset. The primary goal is to uncover potential security risks, operational weaknesses, and performance bottlenecks through a comprehensive analysis of system logs. The Knowledge Discovery in Databases (KDD) process was utilized to manage the complexity of the dataset effectively. This structured methodology enabled efficient data handling, cleaning and preparation. By leveraging the KDD process, the analysis distinguished patterns indicative of normal and abnormal behavior, identified clusters of high-risk events based on suspicious trace of useragents. By combining statistical techniques, machine learning algorithms, and domain expertise, the analysis contributed significantly to the overall enhancement of system monitoring and management.

8.1.1 Special Challenges

- The challenges in this dataset includes efficiently handling the high volume of log entries.
- Additionally, the dataset used in this project was not labeled, which posed difficulties for supervised analysis.

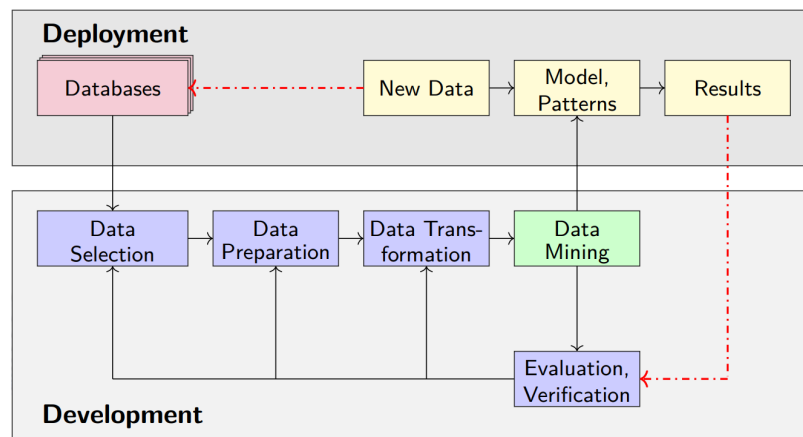


Figure 8.1: KDD Process
[Win24]

- Another challenge was that log entries were not sequentially arranged by **TraceId**, as logs are ordered by timestamp due to concurrent requests. Grouping and organizing requests using **TraceId** was necessary for accurate analysis, adding complexity to preprocessing.

8.2 Database

The database used in this project comprised of log files collected from the system. The analysis was performed collectively on nine log files dated 20th November, 21st November, 23rd November, 24th November, 26th November, 27th November, 29th November, 30th November, and 1st December, 2024.

8.3 Data Selection

Data selection is a critical step that directly influences the project's outcomes. It focuses on identifying relevant portions of the log files for analysis while filtering out irrelevant data to ensure only attributes contributing to the project objectives are prioritized.

8.3.1 Origin

Logs generated by monitoring tools and event management systems, specifically collected from the Graylog server.

8.3.2 Data Format

The file format is `.log`.

8.3.3 Features

The dataset includes log entries with the following attributes

- Timestamps
- PID
- Logger
- Message
- Scope (e.g., TraceId, RequestID)
- Application
- State
- EventID

8.3.4 Size

The dataset is approximately 3.34 GB size.

8.4 Data Preparation

Data preparation in the Knowledge Discovery in Databases (KDD) process, ensures the dataset is clean, consistent and ready for analysis.

8.4.1 Merging and Conversion

- The initial step involved merging all nine individual log files into a single consolidated file.
- Then the merged file, originally in format `.log`, were converted to CSV file to facilitate easier manipulation as shown in the figure 8.2.

8.4.2 Initial Cleaning

- Excluding entries without a valid TraceId that contain generic error or warning messages as shown in figure 8.3.

A	B	C	D	E	F	G	H	I
Level	Timestamp	PID	Logger	Message	Scope	Application	State	EventId
INFO	2024-11-28T10:09:12.7685	3E+05	Microsoft.AspNet	Request starting HTTP/1.1 PATCH http://api.owds.org/ [SpanId: '9cf598563257a011', RequestId: '0HN8E', Name: 'Ofcas.Datasafe.WebApi', Protocol: 'HTTP/1.1', Method: 'PATCH', C [Id: 1, Name: None]				
				Request: Protocol: HTTP/1.1 Method: PATCH Scheme: http PathBase: Path: /api/v1/identities/08DCFF11-2EB0-45C3-80E8-9752A0D7C62 Connection: close Host: api.owds.org User-Agent: Embarcadero URI Client/1.0 Authorization: [Redacted] Content-Type: application/json-patch+json Content-Length: 17 x-forwarded-proto: [Redacted]				
INFO	2024-11-28T10:09:12.7692	3E+05	Microsoft.AspNet	x-forwarded-port: [Redacted]	[SpanId: '9cf598563257a011', RequestId: '0HN8E', Name: 'Ofcas.Datasafe.WebApi', Protocol: 'HTTP/1.1', Method: 'PATCH', S [Id: 1, Name: 'RequestLo			
INFO	2024-11-28T10:09:12.7701	3E+05	System.Net.Http	Start processing HTTP request GET https://id.orgada/ [SpanId: '1fa4aab449c01925', RequestId: '0HN8E', Name: 'Ofcas.Datasafe.WebApi', HttpMethod: 'GET', Uri: 'https://id.orgada', [Id: 100, Name: 'Request				
INFO	2024-11-28T10:09:12.7704	3E+05	System.Net.Http	Sending HTTP request GET https://id.orgada/data/ [SpanId: '1fa4aab449c01925', RequestId: '0HN8E', Name: 'Ofcas.Datasafe.WebApi', HttpMethod: 'GET', Uri: 'https://id.orgada', [Id: 100, Name: 'Request				
INFO	2024-11-28T10:09:12.7930	3E+05	System.Net.Http	Received HTTP response headers after 22.3768ms - [SpanId: '1fa4aab449c01925', RequestId: '0HN8E', Name: 'Ofcas.Datasafe.WebApi', ElapsedMilliseconds: 22.3768, StatusCode [Id: 101, Name: 'Request				
INFO	2024-11-28T10:09:12.7935	3E+05	System.Net.Http	End processing HTTP request after 23.4104ms - 200 [SpanId: '1fa4aab449c01925', RequestId: '0HN8E', Name: 'Ofcas.Datasafe.WebApi', ElapsedMilliseconds: 23.4104, StatusCode [Id: 101, Name: 'Request				
INFO	2024-11-28T10:09:12.8144	3E+05	Microsoft.AspNet	Executing endpoint 'Ofcas.Datasafe.WebApi.Controllers' [SpanId: '9cf598563257a011', ParentId: '0000000', Name: 'Ofcas.Datasafe.WebApi', EndpointName: 'Ofcas.Datasafe.WebApi.Cc', [Id: 0, Name: 'ExecutingE				
INFO	2024-11-28T10:09:12.8146	3E+05	Microsoft.AspNet	Route matched with action = 'UpdateIdentityV1', o [TraceId: '174200ee2dcccac5dd7221c1b3568d7', Name: 'Ofcas.Datasafe.WebApi', RouteData: {action = 'UpdateIdentityV1', [Id: 102, Name: 'Control				
INFO	2024-11-28T10:09:12.8155	3E+05	System.Net.Http	Start processing HTTP request GET https://id.orgada/ [TraceId: '174200ee2dcccac5dd7221c1b3568d7', Name: 'Ofcas.Datasafe.WebApi', HttpMethod: 'GET', Uri: 'https://id.orgada', [Id: 100, Name: 'Request				
INFO	2024-11-28T10:09:12.8158	3E+05	System.Net.Http	Sending HTTP request GET https://id.orgada/data/ [TraceId: '174200ee2dcccac5dd7221c1b3568d7', Name: 'Ofcas.Datasafe.WebApi', HttpMethod: 'GET', Uri: 'https://id.orgada', [Id: 100, Name: 'Request				

Figure 8.2: Converted CSV file

Level	Timestamp	PID	Logger	Message	Scope
ERROR	2024-11-28T18:16:5	668	Microsoft.EntityFrameworkCore	An error occurred using the connection to database 'datasafe' on server '192.168.244.135'.	{'ParentId': '0000000000000000', 'C

Figure 8.3: Error log

8.4.3 Key Attributes

The following were the key parameters that were identified and prioritized for analysis,

- **TraceId**: Used to track individual requests across log entries, enabling the grouping and analysis of entire request flows.
- **HTTP Status Code**: Identifies the result of a request (e.g., success, client errors, or server errors). Codes such as 200, 404 and 500 provide critical insights into system health.
- **Path**: Represents the API endpoint or resource accessed during a request, useful for pinpointing affected components or services.
- **User-Agent**: Provides details about the client or system making the request, helping to identify trends in usage or unusual activity from specific sources.

8.5 Data Transformation

This section focuses on manipulating and reshaping the cleaned data into a structured format ready for analysis. The key transformation includes,

1. Grouping log entries by **TraceId** to reconstruct complete request flows.
2. Parsing relevant fields (**TraceId**, **HTTP Status Code**, **Path**, **User-Agent**) from JSON structures.
3. Transforming each row to represent a unique log entry with all associated fields (e.g., aligning **TraceId** with its corresponding attributes) as shown in figure 8.4.

8 Knowledge Discovery in Databases (KDD) Process

	A	B	C	D
	Trace-id	HTTP Status Code	Path	User Agent
1	00000e83229182560bc00dc08d8d0895	200	/api/v1/nodes/08dd0fba-4f8b-4103-8a95-aed373124a23/ek	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like G
2	0000ca9c0504724ec0352caf2d927e26	204	/api/v1/references/HTTP5:%2F%2FOWDS.ORG%2FAG.KZF4J	Mozilla/5.0 (Linux; Android 14; SM-P620 Build/UP1A.231005.007; wv) AppleWe
3	0000d58f79c6040f625be46853e6143f	200	/api/v1/references/d5731268-72f8-4350-8b42-d44b4fa4efe1/codes	
4	0001693fd167addef160ab0dddfb31d8	200	/api/v1/services/8054f549-6c59-4191-afb3-d31efc46f17e	Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrc
5	0001c4d4a65e06934a9e38f0f71b6496	204	/api/v2/nodes/ebf9e23b-3b78-4c30-9dab-cf963fb01f1a/pro	Mozilla/5.0 (Linux; Android 14; SM-S911B Build/UP1A.231005.007; wv) AppleWe
6	0001d8ca5b776e77d80a9a3ec83903f7d	204	/api/v1/nodes/08dac0aa-8ed5-4ee8-866f-1c22990e0ef0/ele	Mozilla/5.0 (iPhone; CPU iPhone OS 17_4 like Mac OS X) AppleWebKit/605.1.15
7	0001fbbee4d73c2f6a9b28dde44c8e77	200	/api/healthz	curl/8.5.0
8	000273de5951b79ad885cc2de52675be	200	/api/healthz	curl/8.5.0
9	0002b1825f778a4cb80fc65f6e765cc	204	/api/v2/assets/650a0fb5-2e26-44c4-8055-2e7a691e1f9b/nc	Mozilla/5.0 (iPhone; CPU iPhone OS 18_1_1 like Mac OS X) AppleWebKit/605.1.1
10	0002f7cc5075af12e46bd55ee4636d81	200	/api/v1/references/43bdf33e-7709-4e25-bbbb-848e309714bb/codes	
11	000302bc46d6fb65289bc2355604cc8f	200	/api/v2/versions	check_http/v2.4.0 (monitoring-plugins 2.4.0)
12	00030a9aeb3eaa3e54a28873d94e81ea	200	/api/v2/versions	check_http/v2.4.0 (monitoring-plugins 2.4.0)
13	000360463cc19ebcad352f42441055b3	200	/api/v2/versions	check_http/v2.4.0 (monitoring-plugins 2.4.0)
14	000379bd6943567e14425b1985acfe	200	/api/v1/elements/d5fd8aa5-89e1-4f7e-9dbf-0f180ef64ceb/	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like G
15	000439937e3f6ec27d4ee47d56d33d57	200	/api/v1/references/HTTP5:%2F%2FOWDS.ORG%2FAG.M9FB	Mozilla/5.0 (Linux; Android 14; moto g14 Build/UTLB34.102-54-1; wv) AppleWe
16	000472fc41c4d21ae9516f91ec850212	200	/api/v1/elements/81dbd634-7ff4-4bc3-b04c-c7dcca5383d/	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:132.0) Gecko/20100101 Firefox/

Figure 8.4: Result after applying data transformation

8.6 Data Mining

Data mining involves extracting hidden patterns and anomalies from datasets. In this project, clustering and anomaly detection were utilized to uncover system irregularities and potential security threats efficiently.

8.6.1 Feature Extraction

Feature extraction involves converting raw data into a format that can be processed by machine learning algorithms.

- In this project, TF-IDF (Term Frequency-Inverse Document Frequency) Vectorization technique is used, to convert textual data (like the Path attribute in the log data) into numerical features. This allows the model to understand which paths are accessed frequently and which are less common, making it easier to identify unusual patterns.
- TF-IDF vectors were combined with additional numeric attributes such as HTTP status codes, unique paths and user-agents.

8.6.2 Clustering

Clustering was performed using DBSCAN (Density-Based Spatial Clustering of Applications with Noise):

- DBSCAN groups requests that shared similar patterns, such as accessing the same paths with similar HTTP Status Codes and User-Agents were grouped into clusters.
- Requests that did not belong to any cluster were flagged as anomalies (DBSCAN_Cluster = -1).

8.6.3 Anormalities

Along with clustering, Anomaly detection focuses on identifying unusual behaviors that deviate significantly from the norm using Isolation forest.

- The model analyzed the combined features (TF-IDF vectors and numeric attributes) to detect anomalous requests.
- By integrating DBSCAN and Isolation Forest approach, robust anomaly detection was ensured through both density-based and statistical methods.

8.7 Model

The model selection was driven by the need to perform both clustering and anomaly detection effectively. A combination of unsupervised and ensemble-based learning techniques was employed to analyze request patterns and identify anomalies.

8.7.1 DBSCAN

DBSCAN is a density-based clustering algorithm, groups data points that are closely packed together based on a density threshold. Data points far from any dense cluster are treated as "noise" or anomalies.

Key Parameters

- **eps:** Set to 0.5, defining the maximum distance between points to form a cluster.
- **min_samples:** Set to 5, specifying the minimum number of points required to form a cluster.

Outcome

Requests not assigned to any cluster (cluster = -1) were flagged as anomalies.

8.7.2 Isolation Forest

Isolation Forest is an ensemble learning technique designed to detect outliers by isolating data points. It works on the principle that anomalies are easier to isolate because they have unique features.

Key Parameters

- **n_estimators:** Set to 100, specifying the number of decision trees.
- **contamination:** Set to 0.01, indicating the proportion of expected anomalies in the dataset.
- **random_state:** Ensured reproducibility of results with a fixed seed value (42).

Outcome

Requests identified as anomalies were flagged with a value of -1.

8.8 Validation/Verification

Validation and verification are critical steps to ensure the reliability and accuracy of the data mining and anomaly detection process. This section outlines the methods used to validate the models and verify the results.

8.8.1 DBSCAN Validation

The results of the DBSCAN clustering were validated as follows:

- The clustering results were reviewed to ensure that data points in the same cluster exhibited similar patterns.
- Anomalies (`Cluster = -1`) were manually inspected to confirm that they represented requests deviating from normal patterns.

8.8.2 Isolation Forest Validation

The Isolation Forest model's performance was validated using the following methods:

- **Anomaly Score Distribution:** The distribution of anomaly scores was evaluated to differentiate anomalies from regular requests.
- **Manual Inspection:** A subset of flagged anomalies was reviewed to confirm their unusual characteristics.

8.9 Data Visualization

8.10 Conclusion

Bibliography

- [FPS97] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, “Data mining and knowledge discovery in databases: Implications for scientific databases,” *9th International Conference on Scientific and Statistical Database Management*, 1997. DOI: 10.1109/SSDM.1997.621141. [Online]. Available: <https://ieeexplore.ieee.org/document/621141>.
- [MFH00] A. Mockus, R. T. Fielding, and J. Herbsleb. “The kdd process for extracting useful knowledge from volumes of data.” (2000), [Online]. Available: <https://dl.acm.org/doi/10.1145/240455.240464>.
- [Win24] P. E. Wings, *Lecture in data science and analytics: Knowledge discovery in database (kdd) process*, Master in Industrial Informatics, Hochschule Emden/Leer University of Applied Science, 2024.