University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Threat and Weakness Analysis in SimplyTag

ORGADATA COMPANY, LEER

Manoj Selvaraju - 7025649
Vatsal Mahajan - 7025694
Vijay Singh - 7025700

January 15, 2025

Table of Content I

1 Knowledge Discovery in Database (KDD) Process

2 Conclusion and Future Scope

3 Literature

Introduction

About ORGADATA:

- Orgadata is a leading software company, Specializes in solutions for window and door construction, offering products like Logikal.
- Logikal is Orgadata's software that helps users design, calculate, and manage the production of windows and doors efficiently from start to finish.

Why?

- As a student of Industrial Informatics, We have studied how to digitalize products and processes in line with Industry 4.0 principles.
- In today's digital age, safeguarding sensitive information and system integrity is crucial.
- Threat and Weakness Analysis is an essential step in mitigating risks, preventing breaches, and ensuring operational resilience.

Purpose of Analysis

- Exploit exposed endpoints or unauthorized access.
- Detecting vulnerabilities in Orgadata's systems that could be exploited by malicious actors.
- Prevent unauthorized access attempts.
- Improve data security by safeguarding sensitive information of customer and operational data against breaches.
- Build a predictive model using the KDD process to classify threats and evaluate system requests effectively.
- Provide actionable insights to enhance system security and performance.

Challenges we Faced

- Managing and analyzing large-scale logs for meaningful insights.
- Distinguishing between genuine user activities and malicious attempts.
- Handling complex patterns in user behavior and request logs.

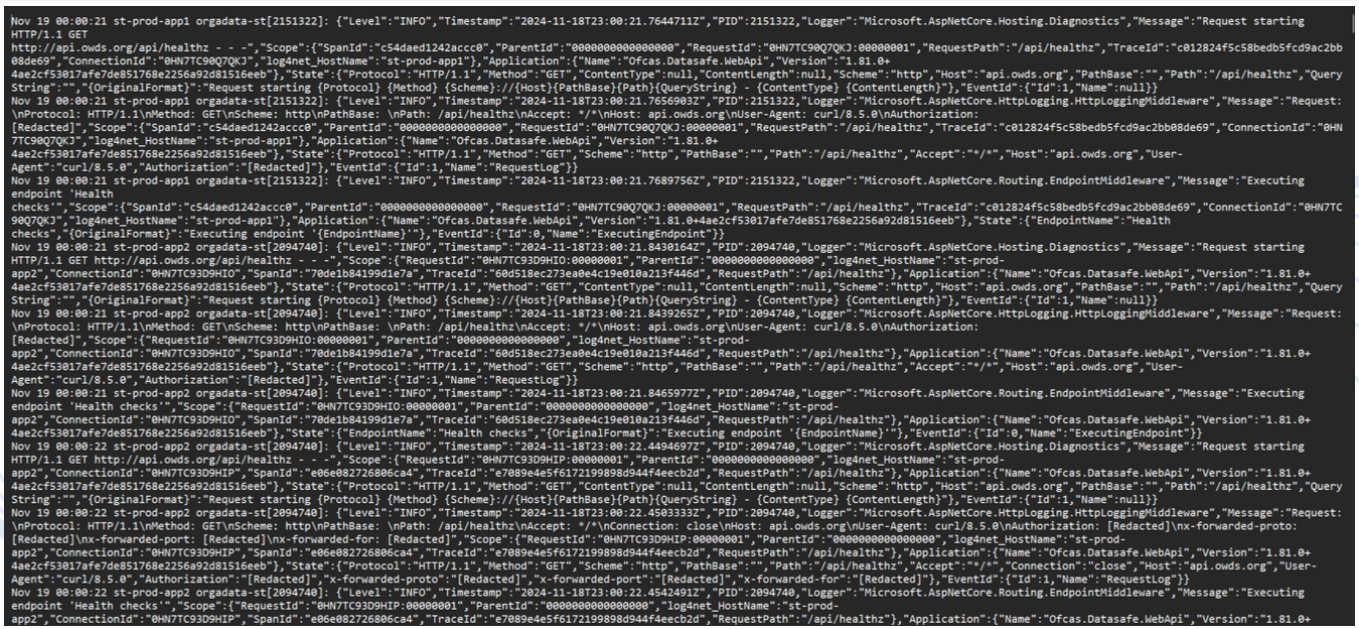

Figure: Raw log data

Knowledge Discovery in Database (KDD) Process

Data Selection:

Identify and Extract relevant data from log files while filtering out irrelevant information.

- **Data Origin:** Logs were sourced from monitoring tools and event management systems, specifically collected via the *Graylog server*.
- **Format:** *.log* format.
- **Size:** 3.34 GB
- **Features:** Timestamps, PID, Logger, Message, Scope (e.g., TraceId, RequestID), Application, State, EventID.

Data Preparation:

Making dataset clean, consistent, and ready for analysis.

- **Merging and Conversion:**
  - Consolidated nine individual log files into a single file.
  - Converted the consolidated file from *.log* format to *CSV* format.
- **Initial Cleaning:**
  - Removed entries which are lacking valid **TraceId**.
  - Excluded error or warning messages.
- **Key Attributes:**
  - **TraceId:** Tracks individual requests across log entries.
  - **HTTP Status Code:** Provides insights into request results:
    - **200:** Successful requests.
    - **404:** Client-side errors (e.g., broken links).
    - **500:** Server-side errors indicating system issues.
  - **Paths:** Represents the API endpoint or resource accessed.
  - **User-Agent** Captures details about the client or system making the request.

Data Preparation:

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.Asp	Request starting HTTP/1.1 PATCH http://api.owds.o	{'SpanId': '9cf398563257a011', 'RequestId': '0HN8E	{'Name': 'Ofcas.Datasafe.WebA	{'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/08DCFF11-2EBD-45C3-8DE8-97552ACD7C62 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.Asp	x-forwarded-port: [Redacted]	{'SpanId': '9cf398563257a011', 'RequestId': '0HN8E	{'Name': 'Ofcas.Datasafe.WebA	{'Protocol': 'HTTP/1.1', 'Method': 'PATCH', 'S	{'Id': 1, 'Name': 'RequestLo
INFO	2024-11-28T10:09:12.7701	3E+05	System.Net.H	Start processing HTTP request GET https://id.orgada	{'SpanId': '1fa4aab449c01925', 'RequestId': '0HN8E	{'Name': 'Ofcas.Datasafe.WebA	{'HttpMethod': 'GET', 'Uri': 'https://id.orgad	{'Id': 100, 'Name': 'Request
INFO	2024-11-28T10:09:12.7704	3E+05	System.Net.H	Sending HTTP request GET https://id.orgadata.com/	{'SpanId': '1fa4aab449c01925', 'RequestId': '0HN8E	{'Name': 'Ofcas.Datasafe.WebA	{'HttpMethod': 'GET', 'Uri': 'https://id.orgad	{'Id': 100, 'Name': 'Request
INFO	2024-11-28T10:09:12.7930	3E+05	System.Net.H	Received HTTP response headers after 22.3768ms - 2	{'SpanId': '1fa4aab449c01925', 'RequestId': '0HN8E	{'Name': 'Ofcas.Datasafe.WebA	{'ElapsedMilliseconds': 22.3768, 'StatusCode	{'Id': 101, 'Name': 'Request
INFO	2024-11-28T10:09:12.7935	3E+05	System.Net.H	End processing HTTP request after 23.4104ms - 200	{'SpanId': '1fa4aab449c01925', 'RequestId': '0HN8E	{'Name': 'Ofcas.Datasafe.WebA	{'ElapsedMilliseconds': 23.4104, 'StatusCode	{'Id': 101, 'Name': 'Request
INFO	2024-11-28T10:09:12.8144	3E+05	Microsoft.Asp	Executing endpoint 'Ofcas.Datasafe.WebApi.Controll	{'SpanId': '9cf398563257a011', 'ParentId': '000000(	{'Name': 'Ofcas.Datasafe.WebA	{'EndpointName': 'Ofcas.Datasafe.WebApi.Cc	{'Id': 0, 'Name': 'ExecutingE
INFO	2024-11-28T10:09:12.8149	3E+05	Microsoft.Asp	Route matched with {action = "UpdateIdentityV1", c	{'TraceId': '17d200ee2dcccace5dd7221c1b3568d7',	{'Name': 'Ofcas.Datasafe.WebA	{'RouteData': '{action = "UpdateIdentityV1",	{'Id': 102, 'Name': 'Controll
INFO	2024-11-28T10:09:12.8155	3E+05	System.Net.H	Start processing HTTP request GET https://id.orgada	{'TraceId': '17d200ee2dcccace5dd7221c1b3568d7',	{'Name': 'Ofcas.Datasafe.WebA	{'HttpMethod': 'GET', 'Uri': 'https://id.orgad	{'Id': 100, 'Name': 'Request
INFO	2024-11-28T10:09:12.8158	3E+05	System.Net.H	Sending HTTP request GET https://id.orgadata.com/	{'TraceId': '17d200ee2dcccace5dd7221c1b3568d7',	{'Name': 'Ofcas.Datasafe.WebA	{'HttpMethod': 'GET', 'Uri': 'https://id.orgad	{'Id': 100, 'Name': 'Request

Figure: 2.CSV File

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

Figure: 3.Error Message

Data Transformation

Restructure and manipulate the cleaned data for analysis.

- Grouped log entries by TraceId to rebuild complete request flows
- Parsed fields from nested JSON structures: TraceId, HTTP Status Code, Path, User-Agent.
- Transformed each log entry into a distinct row, aligning TraceId with its corresponding attributes for seamless analysis.

	A	B	C	D
1	Trace-id	HTTP Status Co	Path	User Agent
2	00000e83229182560bc00dc08d8d0895	200	/api/v1/nodes/08dd0fba-4f8b-4103-8a95-aed373124a23/el	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like G
3	0000ca9c0504724ec0352caf2d927e26	204	/api/v1/references/HTTPS:%2F%2FOWDS.ORG%2FAG.KZF4JI	Mozilla/5.0 (Linux; Android 14; SM-P620 Build/UP1A.231005.007; wv) AppleWeb
4	0000d58f79c6040f625be46853e6143f	200	/api/v1/references/d5731268-72f8-4350-8b42-d44b4fa4efe1/codes	
5	0001693fd167addef160ab0dddfb31d8	200	/api/v1/services/8054f549-6c59-4191-afb3-d31efc46f17e	Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chro
6	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
7	0001d8ca5b76e77d80a9a3ec83903f7d	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
8	0001fbbee4d73c2f6a9b28dde44c8e77	200	/api/healthz	curl/8.5.0
9	000273de5951b79ad885cc2de52675be	200	/api/healthz	curl/8.5.0
10	0002b1825f778a4cb8e0fc65f6e765cc	204	/api/v2/assets/650a0fb5-2e26-44c4-8055-2e7a691e1f9b/no	Mozilla/5.0 (iPhone; CPU iPhone OS 18_1_1 like Mac OS X) AppleWebKit/605.1.1
11	0002f7cc5075af12e46bd55ee4636d81	200	/api/v1/references/43bdf33e-7709-4e25-bbbb-848e309714bb/codes	
12	000302bc46d6fb65289bc2355604cc8f	200	/api/v2/versions	check_http/v2.4.0 (monitoring-plugins 2.4.0)
13	00030a9aeb3eaa3e54a28873d94e81ea	200	/api/v2/versions	check_http/v2.4.0 (monitoring-plugins 2.4.0)
14	000360463cc19ebcad352f42441055b3	200	/api/v2/versions	check_http/v2.4.0 (monitoring-plugins 2.4.0)
15	000379bd694343567e14425b1985acfe	200	/api/v1/elements/d5fd8aa5-89e1-4f7e-9dbf-0f180ef64ceb/	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like G
16	000439937e3f6ec27d4ee47d56d33d57	200	/api/v1/references/HTTPS:%2F%2FOWDS.ORG%2FAG.M9FB	Mozilla/5.0 (Linux; Android 14; moto g14 Build/UTLB34.102-54-1; wv) AppleWeb
17	000472fc41c4d21ae9516f91ec850212	200	/api/v1/elements/81dbd634-7fff-4bc3-b04c-c7dccad5383d/	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:132.0) Gecko/20100101 Firefox/1

Figure: 4.Bar Chart

Data Mining

Feature Extraction

Extracted hidden patterns and anomalies from the selected data.

- **Path-Based Features (Approach 1 & 2):**
  - Extracted key features based on the Path attribute. Features included:
    - **Path length**
    - **Presence of special characters**
    - **SQL keywords**
    - **Path traversal attempts**
    - **Suspicious file extensions**
  - Applied TF-IDF vectorization on the Path attribute to convert API endpoint access into numerical features for machine learning.
- **User-Agent Analysis (Approach 3):**
  - Analyzed User-Agent strings to detect patterns linked to suspicious or malicious activities.

Data Mining

Clustering and Anomaly Detection

- Approach 1: Path and Frequency-Based Clustering
  - Combined extracted path features with frequency metrics and HTTP status codes.
  - Used **DBSCAN** for clustering and **Isolation Forest** for outlier detection.
- Approach 2: TF-IDF and Clustering
  - Utilized TF-IDF vectorized features and numeric attributes for clustering.
  - Integrated **DBSCAN** and **Isolation Forest** for robust anomaly detection, flagging unusual requests.
- Approach 3: User-Agent Pattern Analysis
  - Focused on identifying suspicious behaviors using **User-Agent** patterns.

Data Modelling

Used Unsupervised learning to perform clustering and anomaly detection

Model 1: DBSCAN

- **Purpose:** Group data points based on density; flag points that don't belong to any cluster as anomalies.
- **Key Parameters:**
  - **eps: 0.5**
  - **min_samples: 5**
- **Outcome:** Requests not assigned to any cluster (cluster = -1) were flagged as anomalies.

Model 2: Isolation Forest

- **Purpose:** Detect anomalies by isolating data points, as anomalies are easier to separate from the rest of the data.
- **Key Parameters:**
  - **n_estimators: 100**
  - **contamination: 0.01**
  - **random_state: 42**
- **Outcome:** Anomalous requests were flagged with a value of -1.

Validation/ Verification

Ensure the reliability and accuracy of data mining and anomaly detection processes

DBSCAN Validation

- **Cluster Review:** Verified data points within clusters had similar patterns.
- **Anomaly Inspection:** Manually checked anomalies (Cluster = -1) for normal deviations.

Isolation Forest Validation

- **Anomaly Score Distribution:** Assessed scores to differentiate anomalies from regular requests.
- **Manual Inspection:** Reviewed flagged anomalies to confirm their unusual characteristics.

Data Visualization

- Figure.5 visualizing anomalies identified via Path and TraceId, showing clustering and deviations from normal access patterns.
- FIgure.6 shows analysis of anomalies detected based on User-Agent behaviors

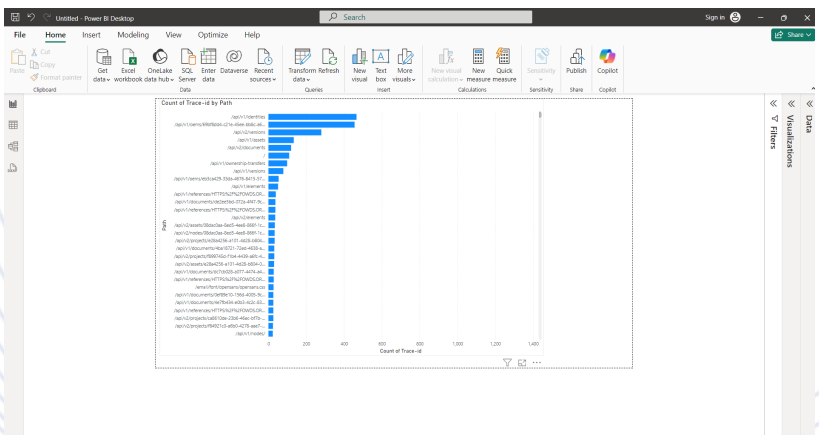

Figure: 5.Bar Chart

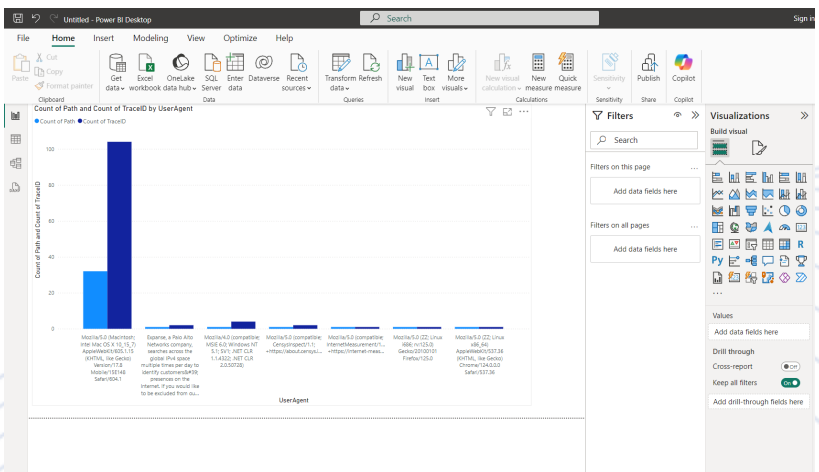

Figure: 6.Bar Chart

.

Conclusion and Future Scope

Conclusion:

- XXX

Future Scope:

- XXX

Literature

- XXX

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Thank you

for your attention