Hochschule Emden/Leer
Department of Electrical Engineering and Computer Science
Industrial Informatics

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

Introduction

About ORGADATA:

- Orgadata is a leading software company, specializing in solutions for window and door digitalization, offering services like SimplyTag.
- SimplyTag is a tool that provides quick and convenient access to construction-related data for specific elements (e.g., windows or doors) using QR codes, enabling on-site professionals to retrieve essential information in real time.

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

- Understanding the huge data (approx. 2.2 million rows) provided in '.log' file and JSON formats.
- Distinguishing between genuine user activities and malicious attempts.
- Handling complex patterns in user behavior and request logs.

Figure: 1.Raw Log Data

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Application Sector

This project belongs to the Industry Domain, specifically for the Digital Tools and Software Solutions domain, with a focus on:

- Threat Detection
- Data Integrity Assurance
- Operational Security Optimizations
- Cybersecurity Measures

Positioning Analytics in Relevant Architecture

The analytics focus on cybersecurity and anomaly detection, aligning with ISO standards for secure data handling and reliable operations.

- **RAMI 4.0:** Positioned in the Information Layer (real-time data collection, processing and analysis for threat detection).
- **IIRA:** Located in the Control Domain (automated threat detection and response) and the Operations Domain (long-term system optimization and strategic planning).

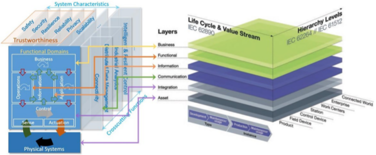

Figure: 2.RAMI_IIRA

Knowledge Discovery in Database (KDD) Process

Data Selection:

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

- **Data Origin:** *Graylog server*
- **Format:** *.log*
- **Size:** 3.34 GB
- **Features:** Timestamps, PID, Logger, Message, Scope (e.g., TraceId, RequestID), Application, State, EventID.

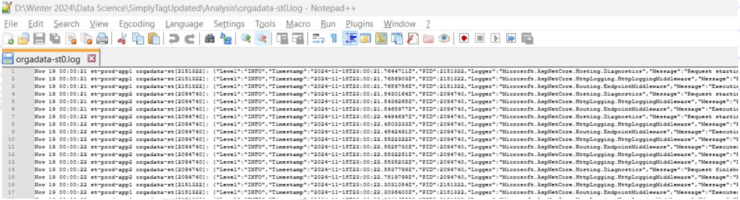

Figure: 3.Database

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Data Preparation:

Given the dataset, the following steps are carryout out to make the dataset clean and ready for analysis

- **Merging and Conversion:** Format .log to CSV
- **Cleaning:** Removed entries with no traceid, errors ,warnings
- **Identified Key Attributes:** Trace-ID, Status code, Path and User Agent

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}
INFO	2024-11-28T10:09:12.7692	3E+05	Microsoft.Asp	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] 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 -	{'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.C	{'Id': 0, 'Name': 'Executing
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: 4.CSV File

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/ele	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/nc	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.M9FBI	Mozilla/5.0 (Linux; Android 14; moto g14 Build/UTLB34.102-54-1; wv) AppleWel
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: 5.Transformed CSV data

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

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.

Validation/ Verification

- **DBSCAN:** Verified datapoints within clusters had similar patterns and the endpoints with SwaggerUI
- **Isolation Forest:** Distribution of anomaly scores was evaluated to differentiate anomalies from regular requests.

Data Visualization

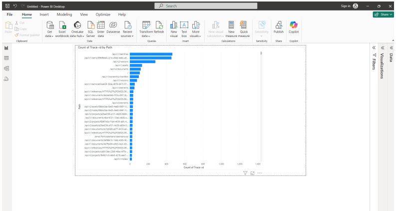

Figure: 6.Analysis based on Path

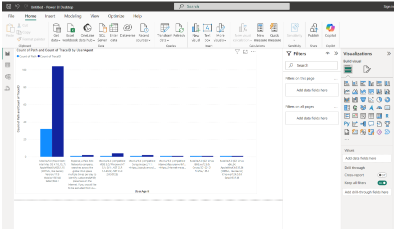

Figure: 7.Analysis based on Useragent

Conclusion and Future Scope

Conclusion:

- Used Knowledge Discovery in Databases (KDD) process to identify potential threats.
- Applied machine learning techniques like DBSCAN and Isolation Forest for clustering and anomaly detection.
- Found exposed endpoints or unauthorized access requests.

Future Scope:

- Unified Approach Integration
- Advanced Model Optimization Algorithms
- Automated Feature Engineering

Thank you

for your attention