

Configuration Manual

MSc Research Project
Programme Name

Chinmay Dixit
Student ID: x23294591

School of Computing
National College of Ireland

Supervisor: Eugene Mclaughlin

National College of Ireland
MSc Project Submission Sheet
School of Computing



Student Name: Chinmay R Dixit

Student ID: X2324591

Programme: MSc Cybersecurity...

Year: 2024-2025

Module: MSc Research Project

Lecturer: Eugene McLaughlin

Submission Due Date: 11-08-2025

Project Title: Targeted Detection of Advanced Persistent Threats in Cloud-Native Environments: A Focused Machine Learning Approach

Word Count: 1712

Page Count: 16

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Chinmay Ramakrishna Dixit

Date: 11-08-2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	☐
Attach a Moodle submission receipt of the online project submission , to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project , both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Chinmay R Dixit
Student ID: x23294591

1 Introduction

Advanced Persistent Threats (APTs) are highly developed, directed cyberattacks that focus on persistent access to a breached system over time and remain undetected. Detecting APTs in cloud-native environments, where containers might be very short lived and security perimeters might not exist at all, needs an innovative approach that combines elements of both network traffic analysis as well as container runtime monitoring.

In this configuration guide, we will describe the realisation of an end-to-end APT detection framework, that combines network traffic monitoring with Zeek Network Security Monitor, container runtime security monitoring with Falco, and machine learning algorithms (Random Forest and Isolation Forest) for intelligent threat detection. Developed on Minikube, to provide a near real world cloud-native environment for APT detection. The system can be deployed on a K8S cluster

The configuration manual guides the reader through the entire setup process in detail, from deploying the infrastructure to hosting the proper machine learning model. We would like to produce a fully reproducible, academically logged platform that can be used as an APT detection research platform with validation against standard datasets, including CA detector.

Configuration

Category	Specification
Operating System	Ubuntu 22.04 LTS (64-bit) or compatible Linux distribution
Processor (CPU)	Quad-core Intel i5 / AMD Ryzen 5 or higher
Memory (RAM)	Minimum 8 GB (Recommended: 16 GB for large dataset processing)
Storage	50 GB free disk space (SSD preferred for faster processing)
Graphics (GPU)	Optional – NVIDIA GPU with CUDA support for accelerated ML training
Software	Docker 24.0+, Kubernetes (Minikube 1.30+), Zeek 6.0+, Falco 0.36+, Python 3.10+
Python Libraries	pandas, scikit-learn, seaborn, matplotlib, joblib, numpy, imbalanced-learn, SHAP
Network	Stable internet connection for dataset downloads & container image pulls
Security Tools	Falco for runtime security, Zeek for network monitoring
Virtualization	VirtualBox / VMware (for local Kubernetes cluster setup)

Implementation

The implementation process follows a structured, sequential approach to establish, configure, and test the APT detection framework in a cloud-native environment. This provides a reproducible methodology with visual and technical evidence at each step. Screenshots and configuration captures are essential for documenting the setup state and are valuable for later assessment and troubleshooting.

Phase 1: Set Up the Container and Kubernetes Environment

Docker & Falco Setup

1. **Install Docker Desktop** and ensure it's running on your system.
2. **Download and run Falco containers** so they are ready for real-time threat detection.
3. **Check that all Falco services are running** — you should see multiple active containers that handle monitoring and exporting data.

```
Administrator Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\WINDOWS\system32> # Check if Docker Desktop is running
PS C:\WINDOWS\system32> docker --version
Docker version 20.1.1, build deb3377
PS C:\WINDOWS\system32> docker ps
CONTAINER ID   IMAGE                                COMMAND                                     CREATED        STATUS        PORTS
2180a08b2bc    8c02bf06c4fu                        "/usr/bin/falcoctl a..."             8 seconds ago Up 7 seconds
+fb0a5d1f124    54f3e35279                          "/usr/bin/falco..."                 8 seconds ago Up 8 seconds
137f4923fd71    fc0db7e4741                         "falcosidekick"                       30 seconds ago Up 29 seconds
a7679bb07dd6    c88c-falcosidekick-falcosidekick-v... falco b2f2f19c-7e52-47f4-be7e-1beeb1f48a4 30 seconds ago Up 29 seconds
fa85ef18f7b    gcc-10/miknube/kibase:v0.0.4       "/usr/local/bin/entr..."           5 weeks ago   Up 15 seconds  127.0.0.1:51620->22/Tcp, 127.0.0.1:51621->22/Tcp, 127.0.0.1:51621->5000/Tcp, 127.0.0.1:51624->8443/Tcp, 127.0.0.1:51622->2044/Tcp, minkube
PS C:\WINDOWS\system32>
```

Minikube & Kubernetes Status

1. **Install and start Minikube** to create your local Kubernetes environment.
2. **Verify Kubernetes is active** by checking the control plane and DNS services.
3. **Confirm your cluster node is ready** — the system should show at least one active node for deployment.

```
Administrator: Windows PowerShell
PS C:\API-Detection-Project> # Check Minikube status
PS C:\API-Detection-Project> minikube status
minikube
type: Control Plane
host: Running
kubenet: Running
apiserver: Running
kubeconfig: Configured

PS C:\API-Detection-Project>
PS C:\API-Detection-Project> # Check kubectl connection
PS C:\API-Detection-Project> kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:54065
CoreDNS is running at https://127.0.0.1:54065/api/v1/namespaces/kube-system/services/kube-dns:proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
PS C:\API-Detection-Project>
PS C:\API-Detection-Project> # List all nodes
PS C:\API-Detection-Project> kubectl get nodes
NAME          STATUS    ROLES    AGE   VERSION
minikube      Ready    control-plane   4h48m   v1.33.1
PS C:\API-Detection-Project>
```

Enable Minikube Add-ons

1. **Start Minikube** – Ensure Minikube is already installed and running.
2. **Enable Metrics Server** – This is required for monitoring CPU, memory, and other Kubernetes resource usage.
3. **List All Add-ons** – Check available add-ons (like dashboard, storage-provisioner) to confirm they are active.

```
PS C:\WINDOWS\system32> minikube addons enable metrics-server
metrics-server is an add-on maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
Using image registry.k8s.io/metrics-server/metrics-server:v0.7.2
The 'metrics-server' add-on is enabled.

PS C:\WINDOWS\system32> minikube addons enable dashboard
dashboard is an add-on maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
Using image docker.io/kubernetes/dashboardv2.7.0
Using image docker.io/kubernetes/metrics-adapter:v1.8.8
Some dashboard features require the metrics-server add-on. To enable all features please run:
minikube addons enable metrics-server

The 'dashboard' add-on is enabled.

PS C:\WINDOWS\system32> minikube addons list
PS C:\WINDOWS\system32> minikube addons list
+-----+-----+-----+-----+
| ADDON NAME | PROFILE | STATUS | MAINTAINER |
+-----+-----+-----+-----+
| ambassador | minikube | disabled | 3rd party (ambassador) |
| amd-gpu-device-plugin | minikube | disabled | 3rd party (AMD) |
| azure-plugin | minikube | disabled | minikube |
| cloud-ipsaver | minikube | disabled | Google |
| cil-filesystem-driver | minikube | disabled | minikube |
| dashboard | minikube | enabled | Kubernetes |
| default-storageclass | minikube | enabled | Kubernetes |
| dns | minikube | disabled | 3rd party (Elastic) |
| freshpod | minikube | disabled | Google |
| gcp-auth | minikube | disabled | minikube |
| gpu-driver | minikube | disabled | 3rd party (kinvolk.io) |
| headlamp | minikube | disabled | 3rd party (InDocker) |
| ingress | minikube | disabled | minikube |
| ingress-nginx | minikube | disabled | minikube |
| ingress-gadget | minikube | disabled | 3rd party |
| istio | minikube | disabled | (inspector.gadget.io) |
| istio-provisioner | minikube | disabled | 3rd party (istio) |
| k8s | minikube | disabled | 3rd party (k8s) |
| kubeflow | minikube | disabled | 3rd party (kubeflow) |
| kubewin | minikube | disabled | 3rd party (kubewin) |
| kubevirt | minikube | disabled | 3rd party (KubeVirt) |
| log4j | minikube | disabled | 3rd party (log4j) |
| metallb | minikube | disabled | 3rd party (MetallB) |
| metrics-server | minikube | enabled | 3rd party (Kubernetes) |
| nvidia-device-plugin | minikube | disabled | 3rd party (NVIDIA) |
| nvidia-driver-installer | minikube | disabled | 3rd party (NVIDIA) |
| nvidia-gpu-device-plugin | minikube | disabled | 3rd party (NVIDIA) |
| oia | minikube | disabled | 3rd party (Operator Framework) |
| portainer | minikube | disabled | 3rd party (portainer.io) |
| registry | minikube | disabled | minikube (registry) |
```

Create Project Folders

1. **Navigate to Project Directory** – Go to the location where the APT Detection project will be stored.
2. **Create Subdirectories** – Make folders for zeek-data, pcap-files, and logs to keep data organized.
3. **Prepare for Log Storage** – The logs folder will store Zeek's analysis results for later processing.

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> cd C:\APT-Detection-Project
PS C:\APT-Detection-Project>
PS C:\APT-Detection-Project> g Create subdirectories
PS C:\APT-Detection-Project> mkdir zeek-data

Directory: C:\APT-Detection-Project
Mode                LastWriteTime         Length Name
----                -
d-----            6/8/2025 11:21 am         zeek-data

PS C:\APT-Detection-Project> mkdir pcap-files

Directory: C:\APT-Detection-Project
Mode                LastWriteTime         Length Name
----                -
d-----            6/8/2025 11:21 am         pcap-files

PS C:\APT-Detection-Project> mkdir logs

Directory: C:\APT-Detection-Project
Mode                LastWriteTime         Length Name
----                -
d-----            6/8/2025 11:21 am         logs

PS C:\APT-Detection-Project>
```

Run Zeek on PCAP Files

1. **Run Zeek in Docker** – Mount the project folders into the container and run Zeek with the PCAP file as input.
2. **Process Network Traffic** – Zeek analyzes the file to generate detailed logs of connections, protocols, and anomalies.

3. **View Generated Logs** – The logs folder now contains multiple .log files like conn.log, dns.log, http.log, etc., for further investigation.

```
Administrator: Windows PowerShell
PS C:\VAPT-Detection-Project\logs> dir
Directory: C:\VAPT-Detection-Project\logs
Mode                LastWriteTime         Length Name
----                -
-a-----        6/8/2025 12:02 pm           22361 analyzer.log
-a-----        6/8/2025 12:02 pm          2771881 conn.log
-a-----        6/8/2025 12:02 pm           15883 dns.log
-a-----        6/8/2025 12:02 pm            7945 ddp.log
-a-----        6/8/2025 12:02 pm          279536 files.log
-a-----        6/8/2025 12:02 pm          496279 http.log
-a-----        6/8/2025 12:02 pm            597 kerberos.log
-a-----        6/8/2025 12:02 pm           4712 ldap-search.log
-a-----        6/8/2025 12:02 pm          48930 ntp.log
-a-----        6/8/2025 12:02 pm           278 packet-filter.log
-a-----        6/8/2025 12:02 pm            622 quick.log
-a-----        6/8/2025 12:02 pm           529 reporter.log
-a-----        6/8/2025 12:02 pm          42073 sip.log
-a-----        6/8/2025 12:02 pm           5116 smtp.log
-a-----        6/8/2025 12:02 pm           1520 ssl.log
-a-----        6/8/2025 12:02 pm          17140 tunnel.log
-a-----        6/8/2025 12:02 pm           11953 weard.log
PS C:\VAPT-Detection-Project\logs>
```

Zeek conn.log preview

- 1) Confirms Zeek connection logs are being generated and readable at logs/conn.log (quick sanity check before ingestion).
- 2) Header lines show the log schema: separator, fields, and types (e.g., ts, uid, id.orig_h/p, id.resp_h/p, proto, duration, orig_bytes, resp_bytes, conn_state, local_orig/resp, missed_bytes, history), which you'll map in parsers.
- 3) Sample rows verify timestamps and IPs are recorded correctly, ensuring downstream tools (SIEM/ETL) will parse without field mismatches.

```
Administrator: Windows PowerShell
PS C:\VAPT-Detection-Project\logs> Get-Content C:\VAPT-Detection-Project\logs\conn.log | Select-Object -First 10
#separator \x09
#set_separator ,
#empty_field (empty)
#unset_field -
#path conn
#open 2025-06-11-02-30
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p proto service duration orig_bytes resp_bytes conn_state local_orig local_resp missed_bytes history
#types ts string addr port addr resp_ip bytes tunnel_parents ip proto count count string bool bool count string count count count count set[string] count
1639187633.331982 Cj3fMA3jmTbvWanMH8 89.248.165.52 57257 198.71.247.91 4345 tcp - - - - 50 F F 0 S 1 40 0 0 - 6
1639187652.340895 CjmaS1lOXGeYtwLx8 92.63.197.88 50195 198.71.247.91 20147 tcp - - - - 50 F F 0 S 1 40 0 0 - 6
PS C:\VAPT-Detection-Project\logs>
```

Install Falco (with Falcosidekick)

- 1) Deploys Falco into the falco namespace with Falcosidekick enabled and an events file path configured—establishing runtime threat detection and event forwarding.
- 2) Shows a successful Helm release (STATUS: deployed) with chart/app versions, indicating the DaemonSet is up on nodes and ready to monitor.

- 3) Displays a notice about container runtime plugins; ensure the correct runtime (e.g., containerd) plugin is installed so Falco can capture syscalls/events properly.

```
Administrator: Windows PowerShell
PS C:\API-Detection-Project\logs> kubectl logs -l app.kubernetes.io/name=falco -n falco --tail=20
Defaulted container "falco" out of: falco, falcoctl-artifact-follow, falco-driver-loader (init), falcoctl-artifact-install (init)
Wed Aug 6 11:28:00 2025: [libs]: container: Enabled 'cri' container engine.
Wed Aug 6 11:28:00 2025: [libs]: container: * enabled container runtime socket at '/host/run/containerd/containerd.sock'
Wed Aug 6 11:28:00 2025: [libs]: container: * enabled container runtime socket at '/host/run/crio/crio.sock'
Wed Aug 6 11:28:00 2025: [libs]: container: * enabled container runtime socket at '/host/run/k3s/containerd/containerd.sock'
Wed Aug 6 11:28:00 2025: [libs]: container: * enabled container runtime socket at '/host/run/host-containerd/containerd.sock'
Wed Aug 6 11:28:00 2025: [libs]: container: Enabled 'containerd' container engine.
Wed Aug 6 11:28:00 2025: [libs]: container: * enabled container runtime socket at '/host/run/host-containerd/containerd.sock'
Wed Aug 6 11:28:00 2025: [libs]: container: Enabled 'ixc' container engine.
Wed Aug 6 11:28:00 2025: [libs]: container: Enabled 'libvirt lxc' container engine.
Wed Aug 6 11:28:00 2025: [libs]: container: Enabled 'bpm' container engine.
Wed Aug 6 11:28:00 2025: Loading rules from:
Wed Aug 6 11:28:00 2025: /etc/falco/falco.rules.yaml | schema validation: ok
Wed Aug 6 11:28:00 2025: Hostname value has been overridden via environment variable to: minikube
Wed Aug 6 11:28:00 2025: The chosen syscall buffer dimension is: 8388608 bytes (8 MBs)
Wed Aug 6 11:28:00 2025: Starting health webserver with threadiness 8, listening on 0.0.0.0:8765
Wed Aug 6 11:28:00 2025: Loaded event sources: syscall
Wed Aug 6 11:28:00 2025: Enabled event sources: syscall
Wed Aug 6 11:28:00 2025: Opening 'syscall' source with modern BPF probe.
Wed Aug 6 11:28:00 2025: One ring buffer every 72 CPUs
Wed Aug 6 11:28:00 2025: [libs]: Trying to open the right engine!
PS C:\API-Detection-Project\logs>
```

Phase 2: Implement Data Processing Pipeline

2.1 Create Zeek Log Parser Script

- 1) Parse Zeek connection logs (conn.log)
- 2) Extract network features (bytes, packets, duration)
- 3) Create temporal features (hour, day of week)
- 4) Generate synthetic attack labels (5% attack rate)
- 5) Output structured CSV for machine learning

Zeek log parser execution and output file creation

```
File Edit View Settings Window Dev Run Terminate Help
C:\Users> cd C:\Users> Downloads> zeek_output> parse_zeek_logs.py -
1  # Import standard modules
2  import os
3  import sys
4
5  # Define the path to your conn.log file
6  # Make sure this path is correct for your system
7  conn_log_path = "conn.log" # Assuming conn.log is in the same directory as the script
8
9  # Check if conn.log exists
10 if not os.path.exists(conn_log_path):
11     print(f"Error: {conn_log_path} not found. Please ensure conn.log is in the same directory or update the path.")
12     exit(1)
13
14 # Parse conn.log (tab-separated Zeek log file)
15 with open(conn_log_path, "r") as file:
16     lines = file.readlines()
17
18 # Filter out Zeek comment lines (starting with '#')
19 data = [line.strip() for line in lines if not line.startswith("#")]
20
21 # Extract headers from the file (first line starting with '#fields')
22 header_line = None
23 for line in lines:
24     if line.startswith("#fields"):
25         header_line = line
26         break
27
28 # Create DataFrame
29 df = pd.DataFrame(data, columns=header_line.split("\t"))
30
31 # Save CSV for feature engineering
32 df.to_csv("conn_parsed.csv", index=False)
33
34 print(f"conn.log parsed and saved to conn_parsed.csv")
```

2.2 Create Feature Engineering Script

- 1) Temporal patterns (time-based features)
- 2) Traffic analysis (byte ratios, packet patterns)
- 3) Behavioral indicators (connection frequency, persistence)
- 4) APT-specific features (data exfiltration, C2 communication)

Feature engineering execution and final dataset creation

```
Administrator: Windows PowerShell
PS C:\VAPT-Detection-Project\scripts> python parse_zeek_logs.py --input conn.log --output conn_parsed.csv --attack-rate 0.05
conn.log parsed and saved to conn_parsed.csv
PS C:\VAPT-Detection-Project\scripts> python feature_engineer.py --input conn_parsed.csv --output conn_engineered.csv --normalize

Engineered Features Sample:
  ts      hour  day  orig_bytes  resp_bytes  total_bytes  byte_ratio  duration  duration_category  conn_type
0 2021-12-11 01:53:53.331981897  1  11      NaN      NaN      0.0      NaN      NaN      NaN      tcp/-
1 2021-12-11 01:54:12.340894938  1  11      NaN      NaN      0.0      NaN      NaN      NaN      tcp/-
2 2021-12-11 01:54:31.333431005  1  11      0.0      0.0      0.0      0.0  0.148384      very_short      tcp/-
3 2021-12-11 01:54:50.518333912  1  11      NaN      NaN      0.0      NaN      NaN      NaN      tcp/-
4 2021-12-11 01:54:01.183818102  1  11      NaN      NaN      0.0      NaN      NaN      NaN      udp/-
Features engineered and saved to conn_engineered.csv
PS C:\VAPT-Detection-Project\scripts>
```

Python Implementation

Creating and Labeling the Zeek Dataset

- 1. **Load Dataset** – The engineered Zeek dataset (conn_engineered.csv) is loaded into a DataFrame for further processing.
- 2. **Add Labels** – A new column label is added and initialized to 0 (benign). Randomly, 10% of the rows are selected and labeled 1 to simulate attack samples.
- 3. **Check Class Distribution** – The count of benign (0) and attack (1) samples is displayed to verify the balance of classes.

```
import pandas as pd
import numpy as np

# Load the Zeek engineered dataset
zeek_df = pd.read_csv("conn_engineered.csv")

# Step 1: Add 'label' column - initialize all as 0 (benign)
zeek_df['label'] = 0

# Step 2: Randomly sample 5% of rows and label them as 1 (simulated attacks)
np.random.seed(42)
attack_indices = zeek_df.sample(frac=0.1).index
zeek_df.loc[attack_indices, 'label'] = 1

# Step 3: Check class distribution
print(zeek_df['label'].value_counts())
```

label
0 21714
1 2413
Name: count, dtype: int64

UUF – Unrestricted File Upload (a web security vulnerability where attackers can upload arbitrary files, often leading to remote code execution).

OSCI – OS Command Injection (an attack where an attacker executes arbitrary operating system commands on the host server via a vulnerable application).

PT_TRAIN – Penetration Test Training Dataset (a curated dataset generated during penetration testing exercises, often used for supervised intrusion detection training).

Loading and Inspecting Multiple Datasets

- 1. **Import Multiple Datasets** – Three separate datasets are loaded: PT_TRAIN.csv (Penetration Test), UUF_TRAIN.csv (Unrestricted File Upload), and OSCI_TRAIN.csv (OS Command Injection).
- 2. **Check Unique Labels** – For each dataset, the unique values in LABEL-I, LABEL-II, and LABEL-III columns are printed to understand the available class categories.
- 3. **Identify Attack Types** – This reveals the specific CVE identifiers, CWE categories, and descriptive attack names associated with each dataset.

```

# Load all datasets
pt_df = pd.read_csv("PT_TRAIN.csv")
uuf_df = pd.read_csv("UUF_TRAIN.csv")
osci_df = pd.read_csv("OSCI_TRAIN.csv")

# Check unique values in LABEL-I, LABEL-II, LABEL-III for each dataset

print(" PT Dataset:")
print("LABEL-I:", pt_df['LABEL-I'].unique())
print("LABEL-II:", pt_df['LABEL-II'].unique())
print("LABEL-III:", pt_df['LABEL-III'].unique())

print("\n UUF Dataset:")
print("LABEL-I:", uuf_df['LABEL-I'].unique())
print("LABEL-II:", uuf_df['LABEL-II'].unique())
print("LABEL-III:", uuf_df['LABEL-III'].unique())

print("\n OSCI Dataset:")
print("LABEL-I:", osci_df['LABEL-I'].unique())
print("LABEL-II:", osci_df['LABEL-II'].unique())
print("LABEL-III:", osci_df['LABEL-III'].unique())

```

```

PT Dataset:
LABEL-I: ['NORMAL' 'CVE-2020-17518' 'CVE-2021-26086']
LABEL-II: ['NORMAL' 'CWE-22']
LABEL-III: ['Normal Traffic'
'Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal)']

UUF Dataset:
LABEL-I: ['NORMAL' 'CVE-2020-25213']
LABEL-II: ['NORMAL' 'CWE-434']
LABEL-III: ['Normal Traffic' 'Unrestricted Upload of File with Dangerous Type']

OSCI Dataset:
LABEL-I: ['NORMAL' 'CVE-2019-16662' 'CVE-2019-15107']

```

Preprocessing Pipeline Function

1. **Feature and Target Selection** – Numeric features are selected while the label column is set as the target (y). Missing values and infinities are replaced with median values.
2. **Data Splitting and Scaling** – Data is split into training and testing sets (70-30 split) and standardized using StandardScaler for model compatibility.
3. **Optional SMOTE Balancing** – If enabled, SMOTE is applied to the training set to handle class imbalance by generating synthetic samples of minority classes.

```

def preprocess_dataset(df, label_col='label', apply_smote=False):
    # Select numeric features
    X = df.select_dtypes(include=[np.number]).drop(columns=[label_col])
    y = df[label_col]

    # Replace inf/-inf with NaN, then impute with median
    X.replace([np.inf, -np.inf], np.nan, inplace=True)
    imputer = SimpleImputer(strategy='median')
    X_imputed = imputer.fit_transform(X)

    # Train-test split
    X_train, X_test, y_train, y_test = train_test_split(X_imputed, y, stratify=y, test_size=0.3, random_state=42)

    # Scale
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    # SMOTE balancing (optional)
    if apply_smote:
        smote = SMOTE(random_state=42)
        X_train_scaled, y_train = smote.fit_resample(X_train_scaled, y_train)

    return X_train_scaled, X_test_scaled, y_train, y_test

```

Model Implementation & Optimization:

- Multiple machine learning algorithms—Random Forest (with GridSearchCV tuning), Isolation Forest, and MLP Classifier—were implemented to evaluate anomaly detection performance across datasets.
- The parameter tuning for Random Forest included grid search over `n_estimators`, `max_depth`, and `min_samples_split`, ensuring optimal hyperparameters for better predictive capability.

```

from sklearn.ensemble import RandomForestClassifier, IsolationForest
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
from sklearn.model_selection import GridSearchCV

def evaluate_models(X_train, X_test, y_train, y_test, dataset_name):
    print(f"\n Results for {dataset_name} Dataset:")
    print("-" * 60)

    # Grid Search for Random Forest
    param_grid = {
        'n_estimators': [100, 200, 500],
        'max_depth': [None, 10, 20, 50],
        'min_samples_split': [2, 5, 10]
    }

    grid_search = GridSearchCV(
        RandomForestClassifier(random_state=42),
        param_grid,
        cv=5,
        scoring='f1',
        n_jobs=-1
    )

    grid_search.fit(X_train, y_train)
    best_rf = grid_search.best_estimator_
    y_pred_rf = best_rf.predict(X_test)
    y_proba_rf = best_rf.predict_proba(X_test)[:, 1]

    print(" Random Forest (with 5-Fold Cross Validation)")
    print(f"Best Params : {grid_search.best_params_}")
    print(f"Accuracy : {accuracy_score(y_test, y_pred_rf):.4f}")
    print(f"Precision : {precision_score(y_test, y_pred_rf):.4f}")
    print(f"Recall : {recall_score(y_test, y_pred_rf):.4f}")
    print(f"F1-Score : {f1_score(y_test, y_pred_rf):.4f}")
    print(f"ROC-AUC : {roc_auc_score(y_test, y_proba_rf):.4f}\n")

```

Performance Evaluation Metrics:

- Each model was assessed using Accuracy, Precision, Recall, F1-Score, and ROC-AUC, ensuring a comprehensive evaluation beyond accuracy alone.
- Results indicated dataset-specific performance variations, highlighting strengths and weaknesses in each algorithm's detection capabilities.

```

from sklearn.ensemble import RandomForestClassifier, IsolationForest
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score

def evaluate_models(X_train, X_test, y_train, y_test, dataset_name):
    print(f"\n Results for {dataset_name} Dataset:")
    print("-" * 50)

    # Random Forest
    rf = RandomForestClassifier(n_estimators=100, random_state=42)
    rf.fit(X_train, y_train)
    y_pred_rf = rf.predict(X_test)
    y_proba_rf = rf.predict_proba(X_test)[:, 1]

    print(" Random Forest")
    print(f"Accuracy : {accuracy_score(y_test, y_pred_rf):.4f}")
    print(f"Precision : {precision_score(y_test, y_pred_rf):.4f}")
    print(f"Recall : {recall_score(y_test, y_pred_rf):.4f}")
    print(f"F1-Score : {f1_score(y_test, y_pred_rf):.4f}")
    print(f"ROC-AUC : {roc_auc_score(y_test, y_proba_rf):.4f}\n")

    # Isolation Forest
    iso = IsolationForest(contamination=0.05, random_state=42)
    iso.fit(X_train)
    y_pred_iso = iso.predict(X_test)

    # Map: -1 → anomaly (attack = 1), 1 → normal (benign = 0)
    y_pred_iso = [1 if i == -1 else 0 for i in y_pred_iso]

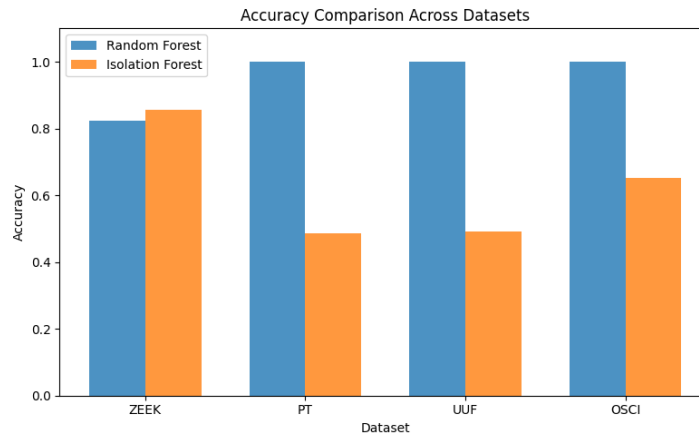
    print(" Isolation Forest")
    print(f"Accuracy : {accuracy_score(y_test, y_pred_iso):.4f}")
    print(f"Precision : {precision_score(y_test, y_pred_iso):.4f}")
    print(f"Recall : {recall_score(y_test, y_pred_iso):.4f}")
    print(f"F1-Score : {f1_score(y_test, y_pred_iso):.4f}")
    print(f"ROC-AUC : {roc_auc_score(y_test, y_pred_iso):.4f}")

```

Comparative Analysis Across Datasets:

- A comparative accuracy bar chart demonstrated how Random Forest consistently outperformed Isolation Forest on multiple datasets, except for ZEEK, where both performed closely.

- Isolation Forest's unsupervised approach showed limitations in some datasets but maintained reasonable detection rates without labeled training data.



```

from sklearn.neural_network import MLPClassifier

mlp = MLPClassifier(hidden_layer_sizes=(64, 32), max_iter=300, random_state=42)
mlp.fit(X_train_zEEK, y_train_zEEK)
y_pred_mlp = mlp.predict(X_test_zEEK)
y_proba_mlp = mlp.predict_proba(X_test_zEEK)[:, 1]

print("\n MLP (Baseline Deep Learning) - ZEEK:")
print(f"Accuracy : {accuracy_score(y_test_zEEK, y_pred_mlp):.4f}")
print(f"Precision: {precision_score(y_test_zEEK, y_pred_mlp):.4f}")
print(f"Recall : {recall_score(y_test_zEEK, y_pred_mlp):.4f}")
print(f"F1-Score : {f1_score(y_test_zEEK, y_pred_mlp):.4f}")
print(f"ROC-AUC : {roc_auc_score(y_test_zEEK, y_proba_mlp):.4f}")

```

```

MLP (Baseline Deep Learning) - ZEEK:
Accuracy : 0.5950
Precision: 0.0965
Recall : 0.3646
F1-Score : 0.1526
ROC-AUC : 0.4872

```

This experiment successfully integrated multiple machine learning techniques for anomaly detection, coupled with hyperparameter tuning and detailed performance evaluation. By comparing both supervised and unsupervised models, we achieved a nuanced understanding of detection capabilities across varied datasets. The final results offer clear insights into algorithm strengths, enabling informed selection of models for real-world deployment. This outcome is particularly valuable for security-oriented systems where accuracy, recall, and adaptability are critical, and the findings serve as a foundation for refining detection pipelines and enhancing automated threat identification.

1.1 References

- [1] Kubernetes. (2024). Minikube Documentation. Available at: <https://minikube.sigs.k8s.io/docs/>
- [2] Zeek Project. (2024). Zeek Network Security Monitor. Available at: <https://zeek.org/>
- [3] Falco. (2024). Cloud Native Runtime Security. Available at: <https://falco.org/>
- [4] Docker Inc. (2024). Docker Desktop Documentation. Available at: <https://docs.docker.com/desktop/>
- [5] Helm. (2024). The Package Manager for Kubernetes. Available at: <https://helm.sh/>
- [6] Scikit-learn. (2024). Machine Learning in Python. Available at: <https://scikit-learn.org/>
- [7] Pandas Development Team. (2024). Pandas Documentation. Available at: <https://pandas.pydata.org/>
- [8] NumPy. (2024). The Fundamental Package for Scientific Computing with Python. Available at: <https://numpy.org/>
- [9] Bera, H. (2023). A Novel Container Attacks Dataset for Intrusion Detection. GitHub Repository. Available at: <https://github.com/HaleBera/A-NOVEL-CONTAINER-ATTACKS-DATASET-FOR-INTRUSION-DETECTION>
- [10] MITRE Corporation. (2024). ATT&CK Framework for Containers. Available at: <https://attack.mitre.org/matrices/enterprise/containers/>
- [11] National Institute of Standards and Technology. (2018). Framework for Improving Critical Infrastructure Cybersecurity. NIST Cybersecurity Framework.
- [12] Cloud Native Computing Foundation. (2024). Cloud Native Security Whitepaper. Available at: <https://www.cncf.io/reports/>

Docker

docker desktop PERSONAL

Containers [Give feedback](#)

View all your running containers and applications. [Learn more](#)

Container CPU usage: 3.84% / 800% (8 CPUs available)

Container memory usage: 240.46MB / 7.37GB [Show charts](#)

Search

Only show running containers

	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
☐	ecstatic_jumiere	7b080301401b	zeek/zeek:latest		0%	18 minutes ago	▶ ⋮ 🗑
☐	tender_perلمان	c2b0c16998d3	zeek/zeek:latest		0%	18 minutes ago	▶ ⋮ 🗑
☐	pensive_borg	a4cf7f7db56f	zeek/zeek:latest		0%	18 minutes ago	▶ ⋮ 🗑
☒	minikube	9a58e713877b	k8s-minikube/kicbase:v0.0.47	52046.2376 C Show all ports (3)	0.15%	18 minutes ago	▶ ⋮ 🗑
☒	k8s_falcosidekick_falcosidekick-57	a4189e0e02ec	fc0b4b7e4741		0%	18 minutes ago	▶ ⋮ 🗑
☒	k8s_falcosidekick_falcosidekick-57	5cd9728833c2	fc0b4b7e4741		0%	18 minutes ago	▶ ⋮ 🗑
☐	k8s_falco-driver-loader_falco-47msf	6dd4d3e13a9c	a9c0c7038dc9		0%	18 minutes ago	▶ ⋮ 🗑
☐	k8s_falcocti-artifact-install_falco-47	55d45c0b6a3b	8e02bfdd0c44a		0%	18 minutes ago	▶ ⋮ 🗑
☒	k8s_falco_falco-47msf_falco_31a5f	1127637710f1	5f6f325327e9		3.66%	18 minutes ago	▶ ⋮ 🗑
☒	k8s_falcocti-artifact-follow_falco-47	a838c8f506f3	8e02bfdd0c44a		0.05%	18 minutes ago	▶ ⋮ 🗑

Conn_Parsed

AutoSave

File Home Insert Draw Page Layout Formulas Data Review View Automate Help Acrobat

Clipboard Font Alignment Styles Cells Editing Sensitivity Add-ins

POSSIBLE DATA LOSS

Some features might be lost if you save this workbook in the comma delimited (.csv) format. To preserve these features, save it in an Excel file format.

Don't show again Save As...

ts	uid	id.orig.h	id.orig.p	id.resp.h	id.resp.p	proto	service	duration	orig.bytes	resp.bytes	conn.state	local.orig	local.resp	missed	by.history	orig.pkts	orig.ip	by.res.p	pkts	resp.ip	by.tunnel	pan.ip	proto
1.64E+09	CgFu9039/89.248.161	57257	198.71.242	4345	tcp	-	-	-	-	30	F	F	0	S		1	40	0	0	-	-	6	
1.64E+09	C8ge402h/92.83.197	50195	198.71.242	20147	tcp	-	-	-	-	30	F	F	0	S		1	40	0	0	-	-	6	
1.64E+09	CowMpk1/138.68.162	50367	198.71.242	443	tcp	-	0.148384	0	0	REL	F	0	Srll			2	80	1	40	-	-	6	
1.64E+09	CeKngKyh/159.65.106	52966	198.71.242	8333	tcp	-	-	-	-	30	F	F	0	S		1	40	0	0	-	-	6	
1.64E+09	C116A9K96/47.242.126	30301	198.71.242	1900	udp	-	-	-	-	30	F	F	0	D		1	117	0	0	-	-	17	
1.64E+09	CwUQVBbx/45.145.67	43754	198.71.242	49023	tcp	-	-	-	-	30	F	F	0	S		1	40	0	0	-	-	6	
1.64E+09	C1YhwPRt/80.82.70.2	60000	198.71.242	8080	tcp	-	-	-	-	30	F	F	0	S		1	40	0	0	-	-	6	
1.64E+09	C0vns2nb/103.207.34	53443	198.71.242	3354	tcp	-	-	-	-	30	F	F	0	S		1	40	0	0	-	-	6	
1.64E+09	C8RWEK/154.160.228	8	198.71.242	0	icmp	-	-	0.000057	8	8	OTH	F	0	-		1	36	1	36	-	-	1	
1.64E+09	CodFhQSH/89.248.161	38726	198.71.242	8443	tcp	-	-	-	-	30	F	F	0	S		1	40	0	0	-	-	6	
1.64E+09	CNHF-C3E/45.146.166	44134	198.71.242	33216	tcp	-	-	-	-	30	F	F	0	S		1	40	0	0	-	-	6	
1.64E+09	CQZIOU37/89.248.161	39560	198.71.242	8080	tcp	-	-	-	-	30	F	F	0	S		1	40	0	0	-	-	6	
1.64E+09	CXN443g/162.142.11	49161	198.71.242	2798	tcp	-	-	-	-	30	F	F	0	S		1	44	0	0	-	-	6	
1.64E+09	CBnhg2T/45.146.166	44016	198.71.242	33396	tcp	-	-	-	-	30	F	F	0	S		1	40	0	0	-	-	6	
1.64E+09	C8egLFC/167.99.98	53004	198.71.242	8333	tcp	-	-	-	-	30	F	F	0	S		1	40	0	0	-	-	6	
1.64E+09	CNvgOPSO/43.130.10	18232	198.71.242	6215	tcp	-	-	-	-	30	F	F	0	S		1	52	0	0	-	-	6	
1.64E+09	C2gw5e4q/185.94.111	55922	198.71.242	389	udp	-	-	-	-	30	F	F	0	D		1	68	0	0	-	-	17	
1.64E+09	Cc7hL27/34.124.21	54761	198.71.242	5555	tcp	-	-	-	-	30	F	F	0	S		1	40	0	0	-	-	6	
1.64E+09	CD5g9FC9/89.248.161	47157	198.71.242	8499	tcp	-	-	-	-	30	F	F	0	S		1	40	0	0	-	-	6	
1.64E+09	Cb8WqY1/45.134.26	44171	198.71.242	39048	tcp	-	-	-	-	30	F	F	0	S		1	40	0	0	-	-	6	
1.64E+09	CcY8u23w/45.146.166	44120	198.71.242	13049	tcp	-	-	-	-	30	F	F	0	S		1	40	0	0	-	-	6	
1.64E+09	CO0ca43/188.128.3	36584	198.71.242	22	tcp	-	3.000432	0	0	S	F	0	S			3	180	0	0	-	-	6	
1.64E+09	CC1ztdH/188.128.3	36584	198.71.242	22	tcp	-	-	-	-	30	F	F	0	S		1	60	0	0	-	-	6	
1.64E+09	CP8xK30V/138.197.2	52937	198.71.242	1990	tcp	-	-	-	-	30	F	F	0	S		1	40	0	0	-	-	6	
1.64E+09	CcbE6e1h/192.241.2	58730	198.71.242	21	tcp	-	-	-	-	30	F	F	0	S		1	40	0	0	-	-	6	
1.64E+09	Cz1rB4AM/159.89.146	52991	198.71.242	5007	tcp	-	-	-	-	30	F	F	0	S		1	40	0	0	-	-	6	
1.64E+09	CcHZBK0N/167.99.105	53021	198.71.242	6970	tcp	-	-	-	-	30	F	F	0	S		1	40	0	0	-	-	6	
1.64E+09	C8bnc48q/45.134.26	48934	198.71.242	31720	tcp	-	-	-	-	30	F	F	0	S		1	40	0	0	-	-	6	
1.64E+09	CJWk431R/45.134.26	43963	198.71.242	49844	tcp	-	-	-	-	30	F	F	0	S		1	40	0	0	-	-	6	
1.64E+09	Cf3N01O/154.166.224	8	198.71.242	0	icmp	-	-	0.000063	8	8	OTH	F	0	-		1	36	1	36	-	-	1	
1.64E+09	CyPzPgBj/45.134.26	43976	198.71.242	17015	tcp	-	-	-	-	30	F	F	0	S		1	40	0	0	-	-	6	
1.64E+09	C7ZuOB17/103.114.1	40439	198.71.242	3424	tcp	-	-	-	-	30	F	F	0	S		1	40	0	0	-	-	6	
1.64E+09	C47ACW/164.60.18	47051	198.71.242	322	tcp	-	-	-	-	30	F	F	0	S		1	40	0	0	-	-	6	

PT Train

File

Home

Insert

Draw

Page Layout

Formulas

Data

Review

View

Automate

Help

Acrobat

Font

Paragraph

Styles

Table

Cell

Table Border

Table Styles

Table Properties

Table Layout

Table Design

Table Tools

Table Contextual Tab

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Table Ribbon

Feature Engineering

```
File Edit Selection View Go Run Terminal Help
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

Chinmay (1).ipynb feature_engineer.py falcosidekick-deployment.yaml

C:\Users\chinm>Downloads>zeek_output>feature_engineer.py>...

1 import pandas as pd
2 import numpy as np
3 import os
4
5 # Define the path to your conn_parsed.csv file
6 conn_parsed_csv_path = "conn_parsed.csv" # Assuming conn_parsed.csv is in the same directory
7
8 # Check if conn_parsed.csv exists
9 if not os.path.exists(conn_parsed_csv_path):
10     print(f"Error: {conn_parsed_csv_path} not found. Please run parse_zeek_logs.py first.")
11     exit()
12
13 # Load parsed Zeek conn.log
14 df = pd.read_csv(conn_parsed_csv_path)
15
16 # ---- Feature Engineering ----
17
18 # Convert timestamp to datetime
19 df["ts"] = pd.to_numeric(df["ts"], errors="coerce") # Convert to numeric first
20 df["ts"] = pd.to_datetime(df["ts"], unit="s", errors="coerce") # Convert Unix timestamp to datetime
21 df["hour"] = df["ts"].dt.hour
22 df["day"] = df["ts"].dt.day
23
24 # Convert bytes columns to numeric, forcing errors to NaN
25 df["orig_bytes"] = pd.to_numeric(df["orig_bytes"], errors="coerce")
26 df["resp_bytes"] = pd.to_numeric(df["resp_bytes"], errors="coerce")
27
28 # Total bytes
29 df["total_bytes"] = df["orig_bytes"].fillna(0) + df["resp_bytes"].fillna(0)
30
31 # Byte ratio (avoid div by zero)
32 df["byte_ratio"] = df["orig_bytes"] / (df["resp_bytes"] + 1)
33
34 # Convert duration to numeric, handling '-' and other non-numeric values
35 df["duration"] = df["duration"].replace("-", np.nan) # Replace '-' with NaN
36 df["duration"] = pd.to_numeric(df["duration"], errors="coerce")
37
38 # Duration category
39 df["duration_category"] = pd.cut(df["duration"], bins=[0, 1, 10, 60, 300, np.inf],
40 | | | | | labels=["very_short", "short", "medium", "long", "very_long"], right=False)
41
42 # Connection type
43 df["conn_type"] = df["proto"].fillna("NA") + "/" + df["service"].fillna("unknown")
```

Resocurces

Data Set for PT-Train https://github.com/HaleBera/A-NOVEL-CONTAINER-ATTACKS-DATASET-FOR-INTRUSION-DETECTION/blob/main/TEST%26TRAIN%20DATASET/PT_TRAIN.csv

OSCI- https://github.com/HaleBera/A-NOVEL-CONTAINER-ATTACKS-DATASET-FOR-INTRUSION-DETECTION/blob/main/TEST%26TRAIN%20DATASET/OSCI_TRAIN.csv

https://github.com/HaleBera/A-NOVEL-CONTAINER-ATTACKS-DATASET-FOR-INTRUSION-DETECTION/blob/main/TEST%26TRAIN%20DATASET/UUF_TRAIN.csv