National College of Ireland

Configuration Manual

MSc Research Project
MSc in Cloud Computing

Vikas Duvva
Student ID: X23380322

School of Computing
National College of Ireland

Supervisor: Ahmed Hamza Ibrahim

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Vikas Duvva
Student ID:	x23380322
Programme:	MSc in Cloud Computing **Year:** 2025-2026
Module:	MSc Research Project
Supervisor:	Ahmed Hamza Ibrahim
Submission Due Date:	28-01-2026
Project Title:	Self-Organizing Cybersecurity Systems with Formal Verification Constraints using Multi-Agent Reinforcement Learning
Word Count:	985 **Page Count:** 6

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: Vikas Duvva

Date: 28-01-2026

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

Vikas Duvva
X23380322

1 Introduction

This guide will describe how to set up and operate the MARL-STL-BFT system for cybersecurity threat detection. The project uses three independent LSTM-PPO agents which collaborate with each other to vote in a majority of 2/3 and detect security threats. The system is proven on the DARPA Transparent Computing dataset, and it can be deployed as a REST API on AWS.
What you'll learn:
- Installing Python and its dependencies
- Training the MARL agents with a pre-trained encoder.
- Related with testing and inference.
- Implementing the system to AWS EC2.

2 System Requirements

- Windows/Ubuntu/Mac
- Python 3.10+
- CPU with min. 2 cores
- 8GB Min RAM
- 10GB Free Space

For Model Training, Google Colaboratory with NVIDIA T4 or higher GPU

3 Software Dependencies

Create a requirements.txt:

torch==2.9.1
torchvision==0.24.
numpy==2.2.6
pandas==2.3.3
scikit-learn==1.7.2
scipy==1.15.3
matplotlib==3.10.7
seaborn==0.13.2
requests==2.32.5
tqdm==4.67.1

boto3==1.41.4
flask==3.0+

4 Project Structure

```
marl-thesis/
├── test_api_comprehensive.py         # MAIN evaluation
├── test_deployed_api_simple.py       # Quick test
├── requirements.txt                  # Python dependencies
├── test_ready.pkl                    # Test dataset (75,001 samples)
├── scaler.pkl                        # Feature normalizer
├── deployment/
│   ├── app.py                # Flask API server
│   ├── models/                       # Trained agent models
├── marl_stl_bft_results/
        └── models/final_marl_agents/
            ├── agent_0.pt            # Permissive Agent 0
            ├── agent_1.pt            # Restrictive Agent 1
            ├── agent_2.pt            # Permissive Agent 2
            ├── scaler.pkl            # Feature scaler
            └── training_stats.json   # Training metrics
```

5 Local Setup

Create Python Environment

```
$ python3 -m venv venv
$ source venv/bin/activate       # On macOS/Linux
$ venv\Scripts\activate          # On Windows
```

6 Data Preparation

The project uses the DARPA Transparent Computing dataset.
Dataset Location - The preprocessed test data is located at ./test_ready.pkl (75,001 samples - balanced test set)

Data Format - 7 normalized features:
1. source_bytes
2. destination_bytes
3. duration
4. source_packets
5. destination_packets
6. source_ttl
7. destination_ttl

Labels: 0 (benign) or 1 (malicious)

7 Saved Model Files from Training

- ./marl_stl_bft_results/models/final_marl_agents/agent_0.pt (5.2 MB)
- ./marl_stl_bft_results/models/final_marl_agents/agent_1.pt (5.2 MB)
- ./marl_stl_bft_results/models/final_marl_agents/agent_2.pt (5.2 MB)
- ./scaler.pkl (StandardScaler for normalization)
- ./training_stats.json (metrics history)

8 Model Inference and Testing

8.1 Quick API test

$ python test_deployed_api_simple.py

To verify the API working after deployment.

8.2 Main Evaluation Test

$ python test_api_comprehensive.py

MAIN evaluation script that tests 500 balanced samples.

It provides complete metrics:
- Accuracy, Precision, Recall, F1 Score
- All 4 actions: Allow, Block, Alert, Quarantine)
- Individual Agent Vote Patterns
- Latency Statistics
- Confidence Scores
- Class-Specific Results
- Example Predictions

9 AWS Deployment

9.1 Prerequisites

- AWS account
- AWS CLI installed
- EC2 key pair created

9.2 EC2 Instance Creation

1. Log in to AWS Console
2. Navigate to EC2 > Launch Instance
3. Choose AMI: Amazon Linux 2023 or Ubuntu 22.04
4. Instance type: t3.medium (2 vCPU, 4 GB RAM)
5. Configure Security Group:
   - Allow SSH (port 22) from your IP
   - Allow HTTP (port 5000) from anywhere
6. Select your key pair
7. Launch instance

9.3 Connect to EC2

$ ssh -i "your-key.pem" ec2-user@INSTANCE_PUBLIC_IP

9.4 Application Deployment

$ python3 -m venv venv
$ source venv/bin/activate
$ pip install -r requirements.txt
$ pip install flask gunicorn

9.5 Copy Model Files to EC2

$ scp -i "your-key.pem" trained_models/*.pt ec2-user@YOUR_PUBLIC_IP:~/deployment/models/

$ scp -i "your-key.pem" scaler.pkl ec2-user@YOUR_PUBLIC_IP:~/deployment/models/

9.6 Start API server

On EC2:
$ cd deployment
$ python app.py

9.7 Test Deployment

Run from local machine:

$ curl http://INSTANCE_PUBLIC_IP:5000/health

Expected response:

{
  "status": "healthy",
  "models_loaded": true,
  "version": "fixed_v2"
}

9.8 Run Evaluation

$ nano test_api_comprehensive.py
# Change: API_URL = http://INSTANCE_PUBLIC_IP:5000
$ python test_api_comprehensive.py | tee test_results.log

This will test 500 samples and generate detailed metrics from the deployment.

10 API Endpoints

10.1 Health Check

GET /health - return system health and model availability status

10.2 Predict Threat

POST /predict
Content-Type: application/json
{
  "features": [0.23, -0.35, 2.67, 1.89, -1.56, 1.34, 0.78]
}

Response:
{
  "prediction": "malicious",
  "action": "Block",
  "confidence": 0.67,
  "vote_count": 3,
  "individual_votes": ["Alert", "Block", "Alert"]
}

11 Performance Benchmarks

Sample Run Performance metrics

Parameter	**Value**
Prediction	malicious
Action	Block
Confidence	0.67
Vote Count	3
Individual Votes	Alert, Block, Alert
Threat Detection Rate	99.20%
False Positive Rate	75.20%
Accuracy	62.00%
Precision	56.88%
F1 Score	0.7230
Median Latency	479 ms
Throughput	2.0 pred/sec

12 References

DARPA Transparent Computing Dataset (2018). Defense Advanced Research Projects Agency, Transparent Computing Engagement 3.
URL: https://github.com/darpa-i2o/Transparent-Computing

Getting Started with AWS EC2 – https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

Flask Web Development Quick Start Guide - https://flask.palletsprojects.com/en/stable/quickstart/