Final Presentation

INFAST: Intelligent Automated Functional and Security Testing activity

European Space Agency

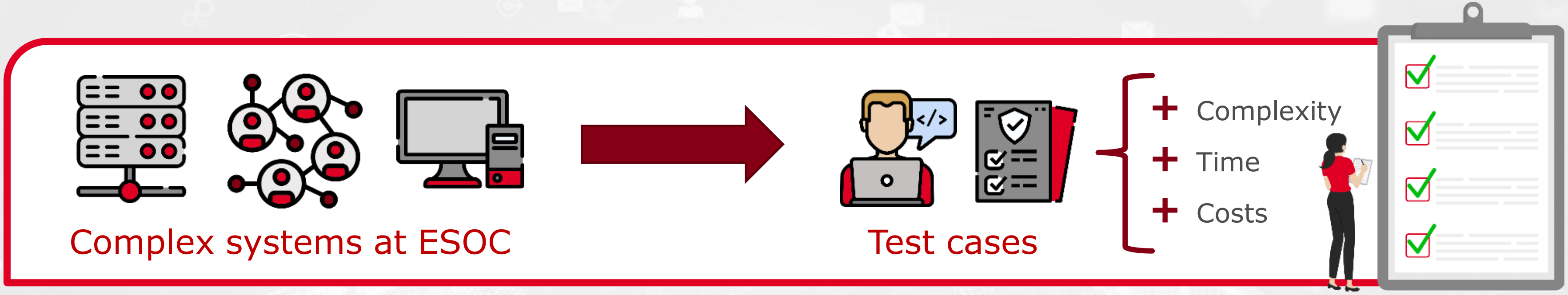
MEMBER OF FEV GROUP

INNOVATING SOLUTIONS

Index

Introduction

Towards INtelligent automated Functional and Security Testing (INFAST)



- INFAST is a project funded by ESA aimed to provide solutions enhanced by AI driven algorithms to automate testing tasks.
- ESA funded TDE (Technology Development Element) activity.
- Two Proof-of-Concept use cases
  - **Root Cause Analysis for Functional testing**
  - **Action Selection for Automated Pentesting**

Root Cause Analysis for Functional testing

Use Case Definition

Tester/ Operator

Execute Test Cases → Analyse Test Failures → Open a new SPR → Root Cause Analysis → Fix error → Close SPR

Import test reports and SPRs

INFAST

API

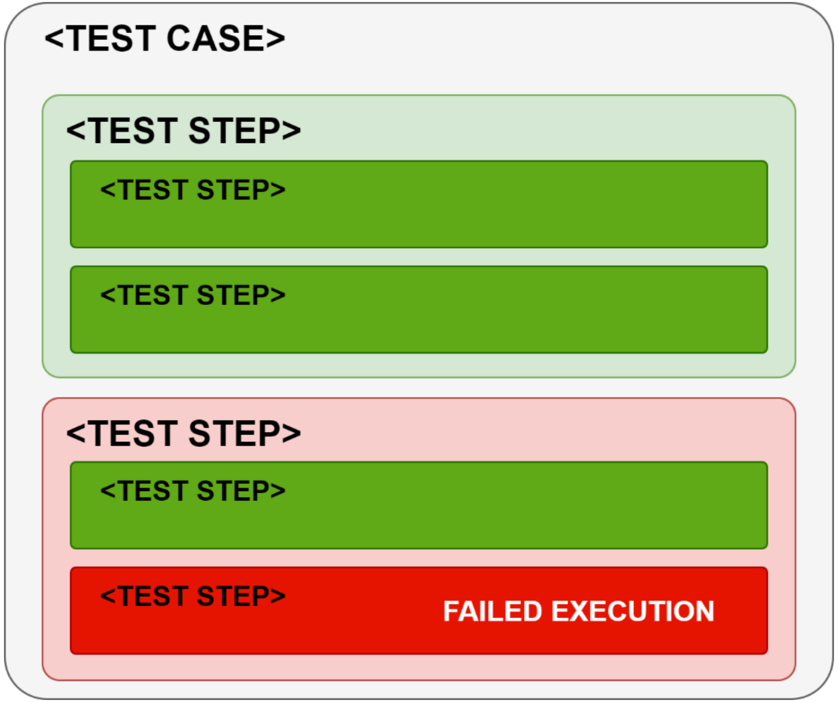
If no related SPRs…

Existing groups of related test failures from different projects and test cases

The tool aims to automate repetitive tasks and serve as an aid to the user in finding possible root causes

Root Cause Analysis for Functional testing

Use Case Definition



GOALS

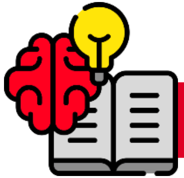
- Root Cause Analysis
- Error Correlation

DATA

- Test Reports
- Software Problem Reports (SPRs)

Root Cause Analysis for Functional testing

Use Case Definition



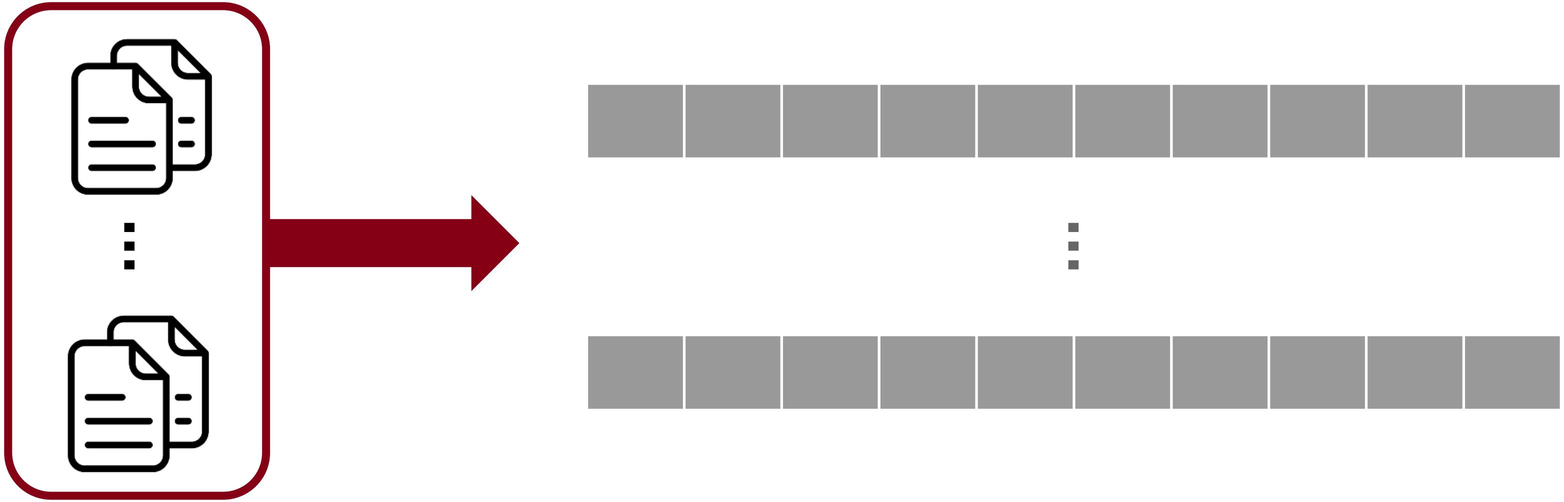
Natural Language Processing (NLP) approach

- **Extract failures from Test Case.**
- **Extract text fields from data sources:**
  - From test failures: Test step description and execution log.
  - From SPRs: SPR description.
- **Obtain relationships based on text similarities.**

Root Cause Analysis for Functional testing

Proposed approach: Data Encoding

- Representation Learning (sentence embeddings).
- **Transformer** (Attention-based) pre-trained model to projects sentences into a 768-dimensional latent space.



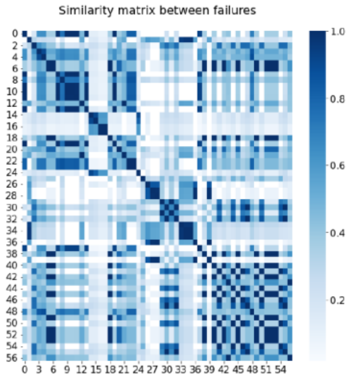
Root Cause Analysis for Functional testing

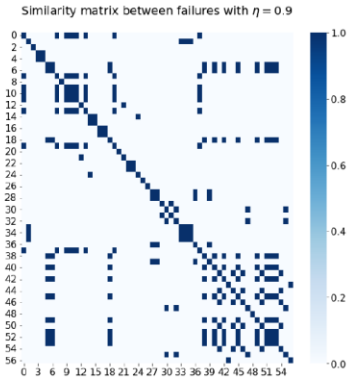
Proposed approach: Similarity

- Compute the distance between vectors using the cosine similarity:

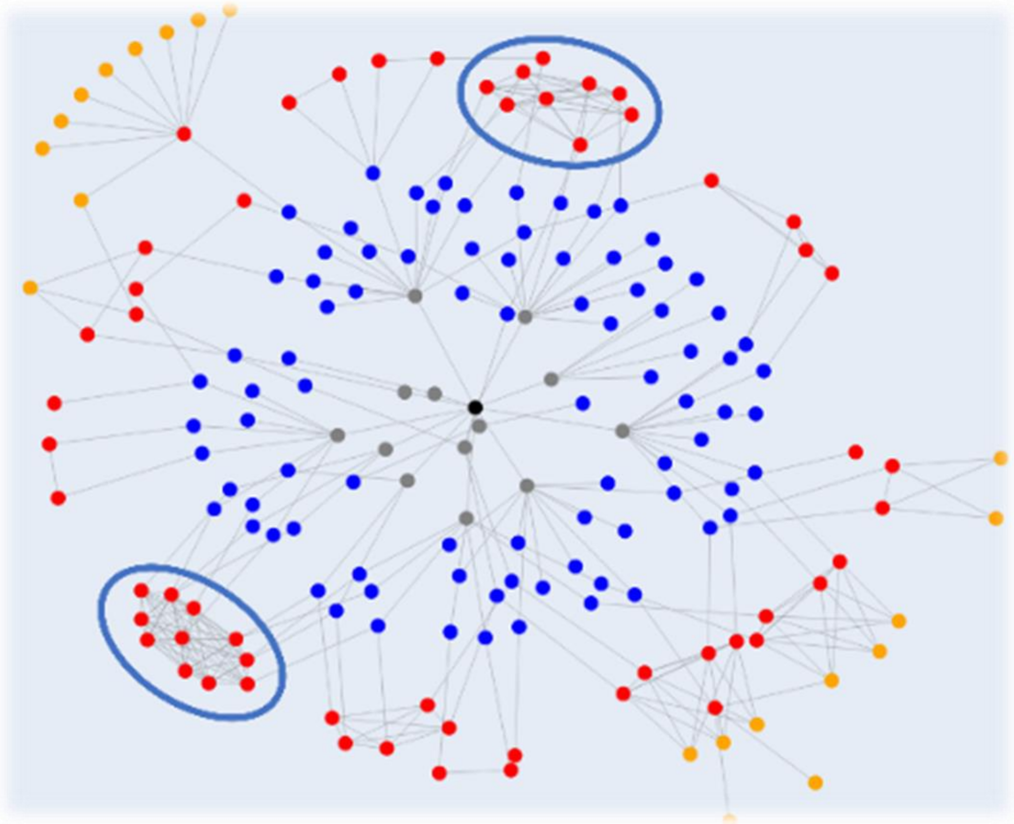
$$S_C(\boldsymbol{x},\boldsymbol{y}) = \frac{\boldsymbol{x}\cdot\boldsymbol{y}}{\|\boldsymbol{x}\|\cdot\|\boldsymbol{y}\|} = \frac{\sum_{i=1}^{n}\boldsymbol{x}_i\boldsymbol{y}_i}{\sqrt{\sum_{i=1}^{n}(x_i)^2}\sqrt{\sum_{i=1}^{n}(y_i)^2}}$$

- Apply Geometric Mean (descriptions/logs) and a threshold to decide:





Root Cause Analysis for Functional testing

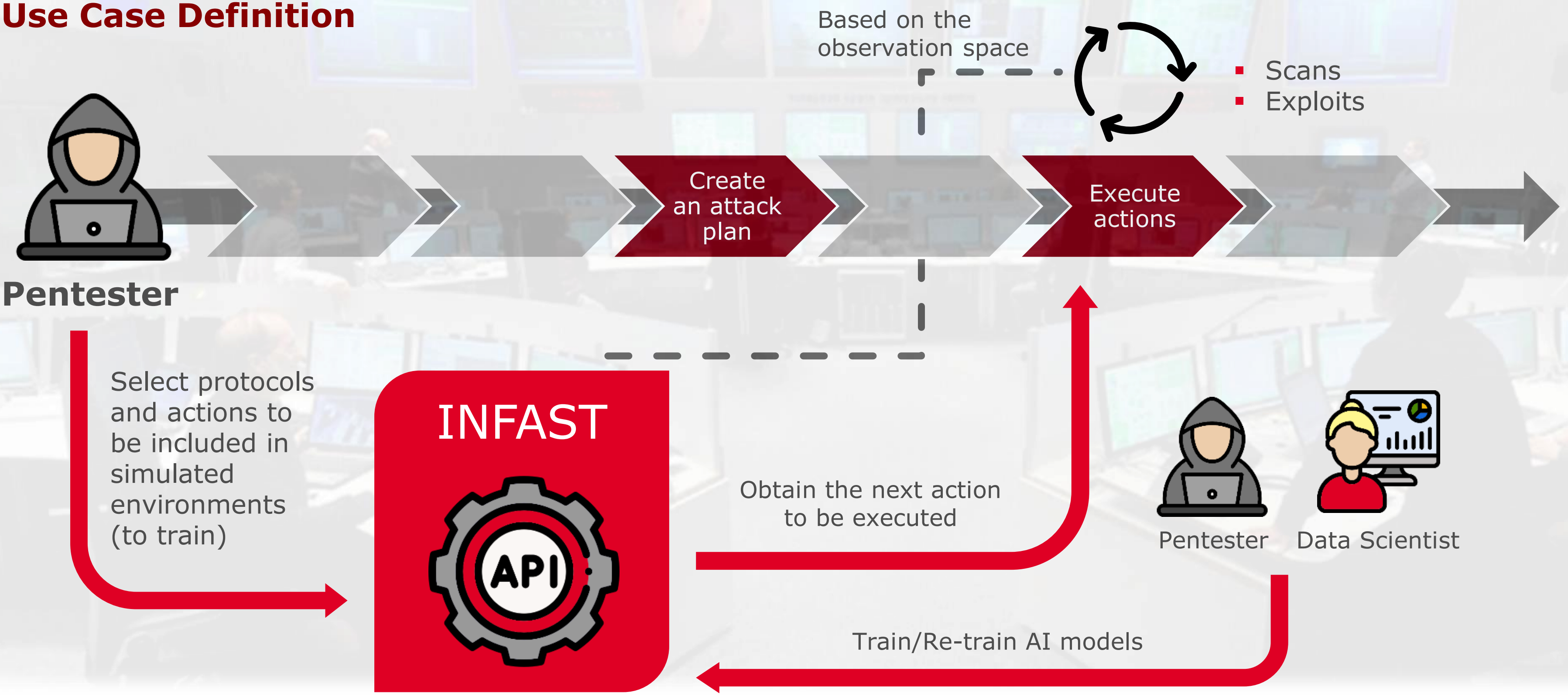


Results

- ✓ Compared different techniques: **Transformers**, TF-IDF, and BM-25.
- ✓ Discovered relationships between test failures belonging to different test projects.
- ✓ Identified **clusters of test failures**: Possible Root Cause.
- ✓ Few relationships between test failures and SPRs.

Action Selection for Automated Pentesting

Use Case Definition



Action Selection for Automated Pentesting

Use Case Definition

Source: European Space Agency (ESA)

- **PenBox** (previously developed by ESA/ESOC) is a proof of concept for penetration test automation:
  - It requires the definition of a scenario or plan of attack.
  - Executes all defined actions in a set order.

GOALS

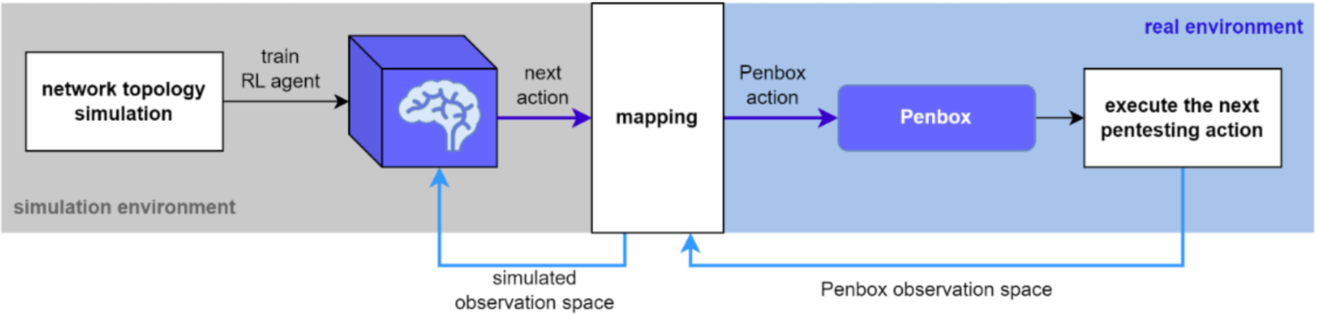
- Automate the selection of the next action to be executed.

Action Selection for Automated Pentesting

Use Case Definition

Reinforcement Learning (RL) approach

- Generate simulated environments (network topologies).
- Train a RL agent based using simulated environments.
- Integrate with Penbox to be executed in real environments.



Action Selection for Automated Pentesting

Simulated Environments Generation

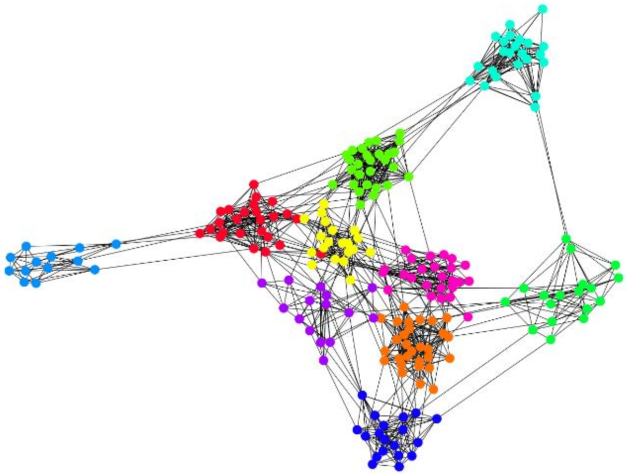
- Simulate environment based on CyberBattleSim:
  - ✓ As realistic as possible
- **Generate Environment**

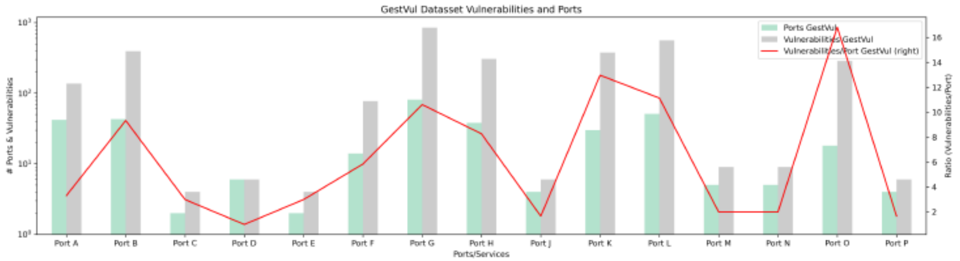
  Step 1 — Generate Global Identifiers

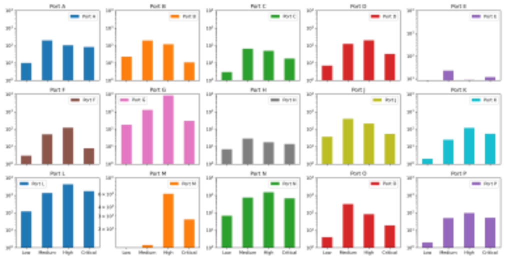
  Step 2 — Define Vulnerabilities

  Step 3 — Generate Random Network Traffic

  Step 4 — Define Nodes







Action Selection for Automated Pentesting

RL Policy: Architectures

- Multi-Layer Perceptron (MLP)
- **Graph Neural Networks (GNNs)**
- Recurrent & Attention Layers

RL Algorithms: Architectures

- On-Policy (PPO, A2C)
- **Off-Policy (DQN)**

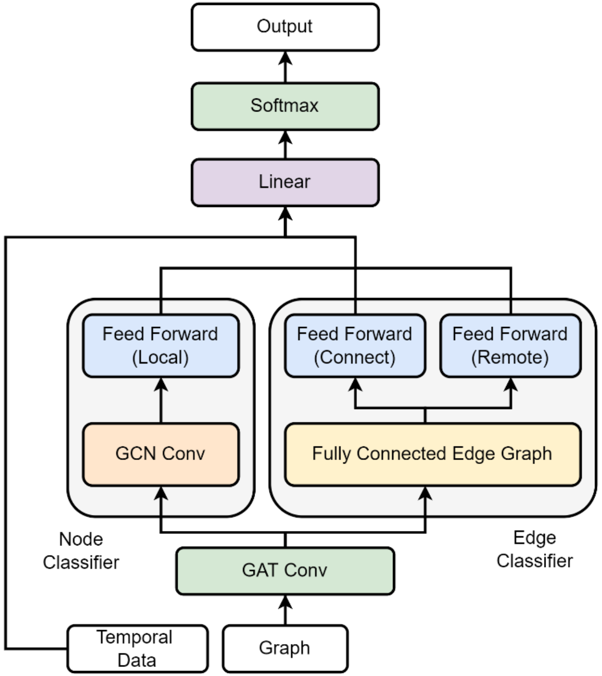
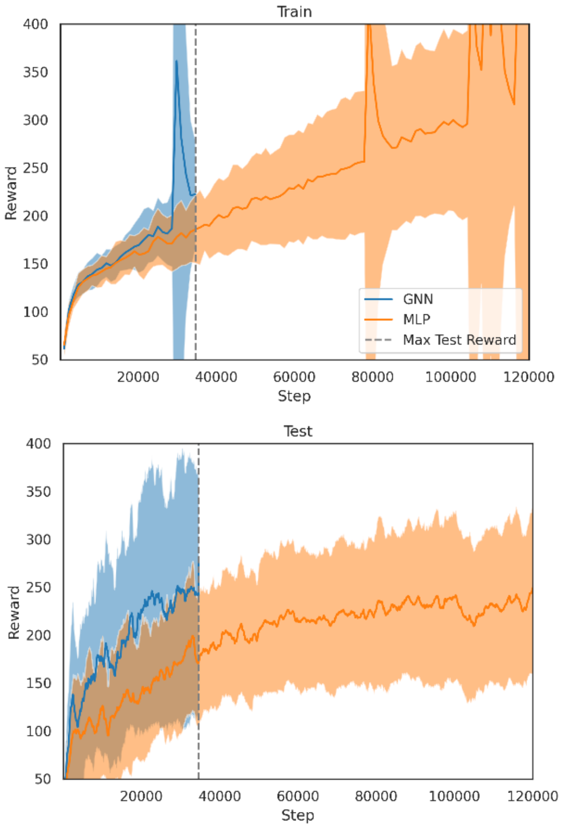


Figure: GNN policy architecture diagram.

Action Selection for Automated Pentesting

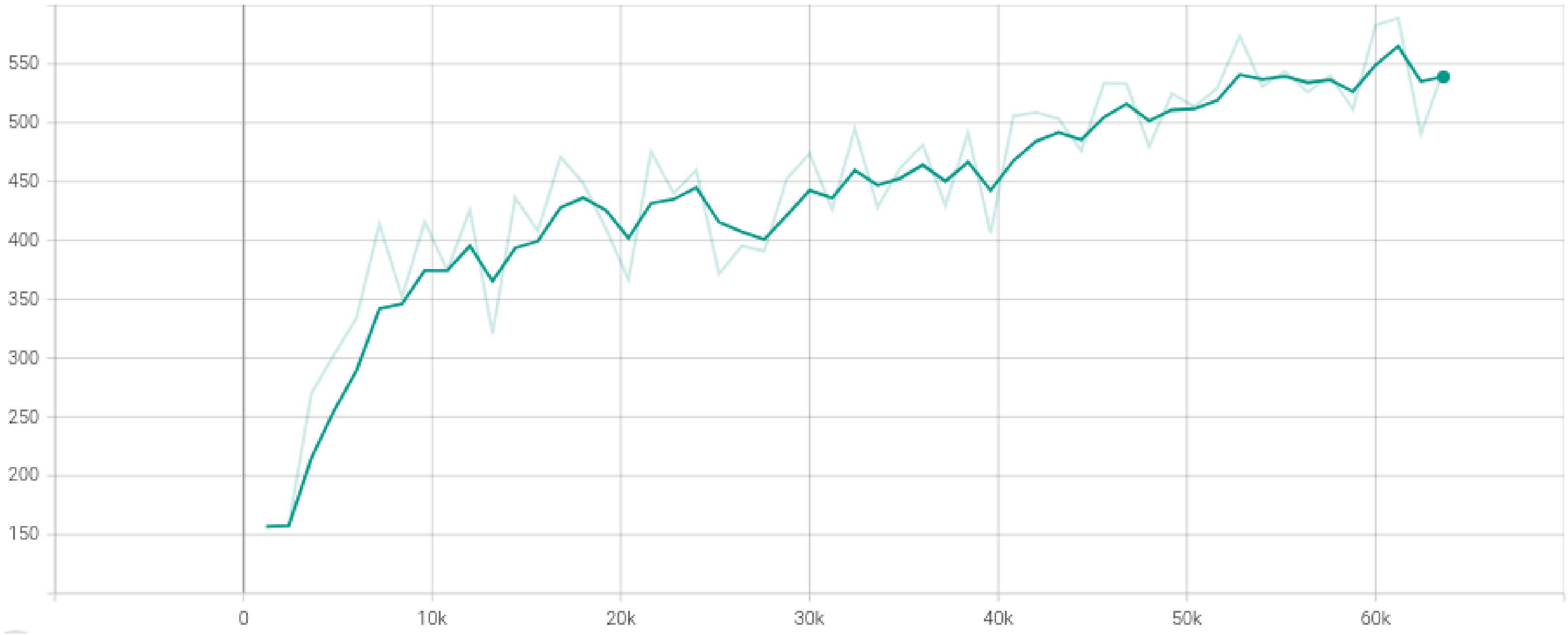
Algorithms/architectures comparison

- Agent achieves converges in both training and testing using DQN.
- GNNs…
  - …outperforms MLP over the testing step
  - Experimentally: Generalizes better achieving maximum test reward
  - Theoretically:
    - Permutation equivariance
    - Non-fixed size input & scalability



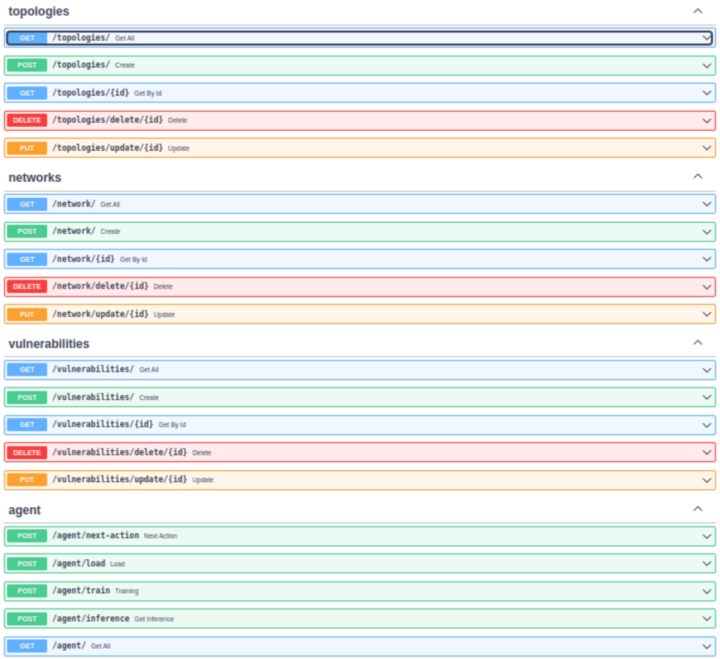
Action Selection for Automated Pentesting

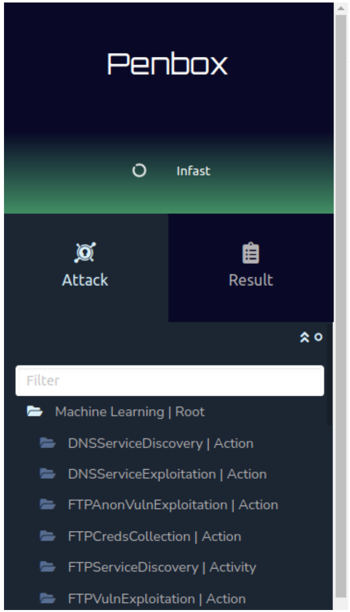
RL agent training

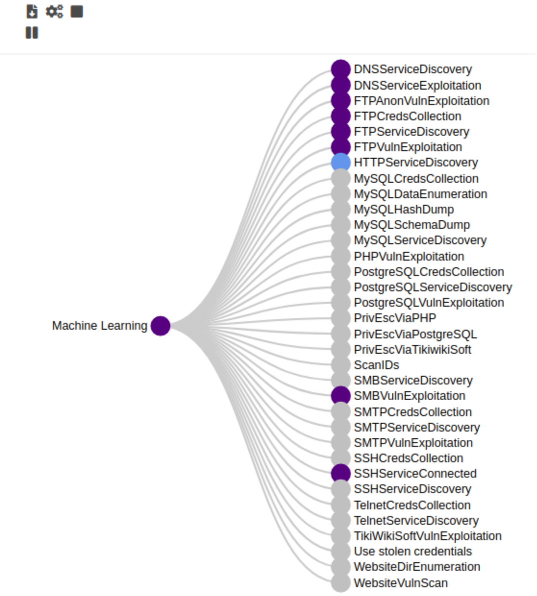


Action Selection for Automated Pentesting

Agent execution in a real Environment (Sim2Real): PenBox

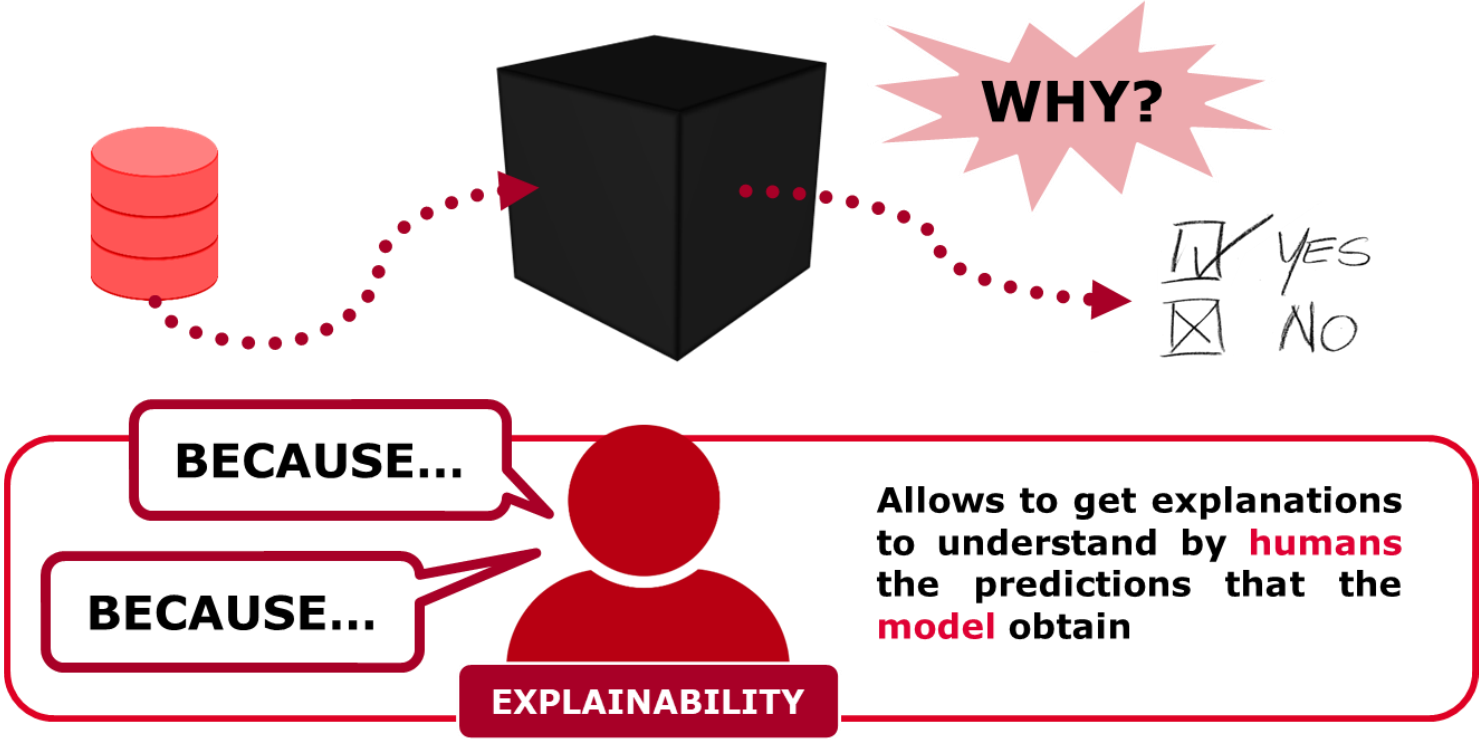






Explainable Artificial Intelligence

Context

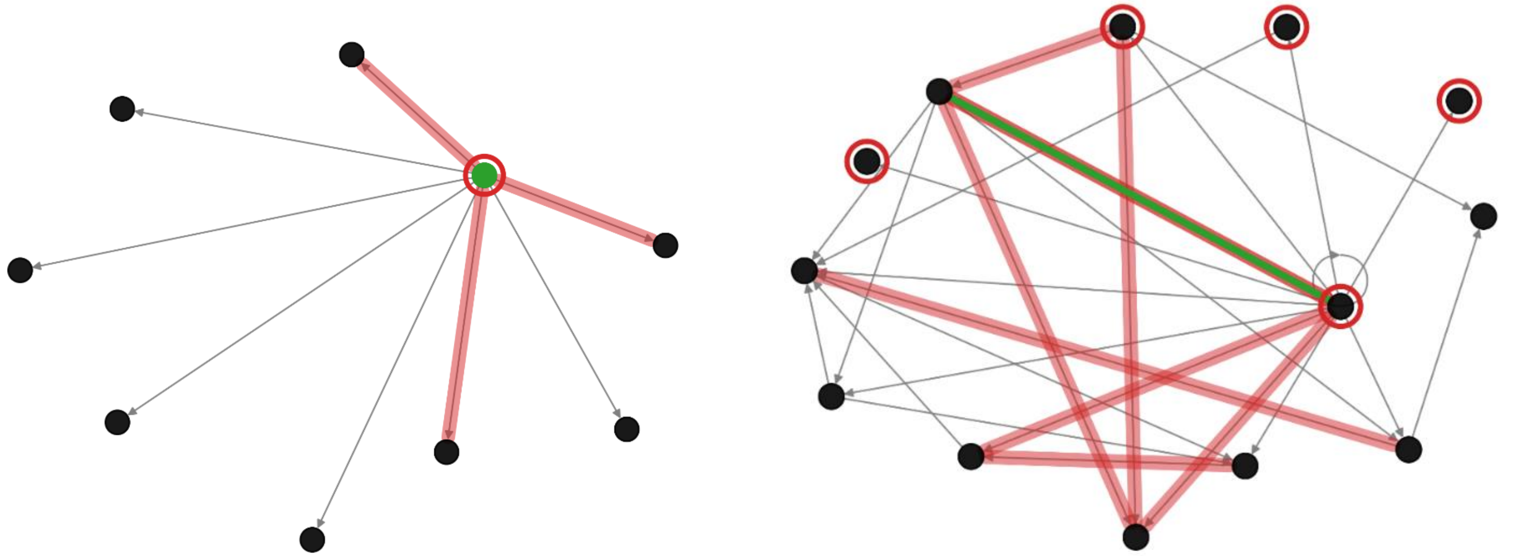


Allows to get explanations to understand by humans the predictions that the model obtain

Explainable Artificial Intelligence

XAI for Automated Pentesting

- **RL + GNN approach: Integrated Gradients algorithm for XAI**
  - The red lines (edge features) and red dots (node feature) denote the importance attributed for a single chosen action (green).



Conclusions and further work

Root Cause Analysis for Functional testing

- ✓ NLP techniques are a suitable approach to obtain similarity.
- ✓ This has allowed failures to be clustered and possible root causes to be identified.
- ✓ Difficulties in relating texts written in different registers.

NEXT STEPS

- Standarise the way the two texts are generated in the future.
- Explore other techniques:
  - **Automatic question-answering**.
  - **Named Entity Recognition and regular expressions**
  - **Bayesian root cause identification techniques.**
- Use of additional data sources such as SUT logs, test specifications and/or repository commit history.

Conclusions and further work

Root Cause Analysis for Functional testing

- ✓ Simulate environments based on CyberBattleSim using real data.
- ✓ Train multiple agents using RL and different architectures.
- ✓ High performance on train and test sets.
- ✓ Use of XAI techniques.

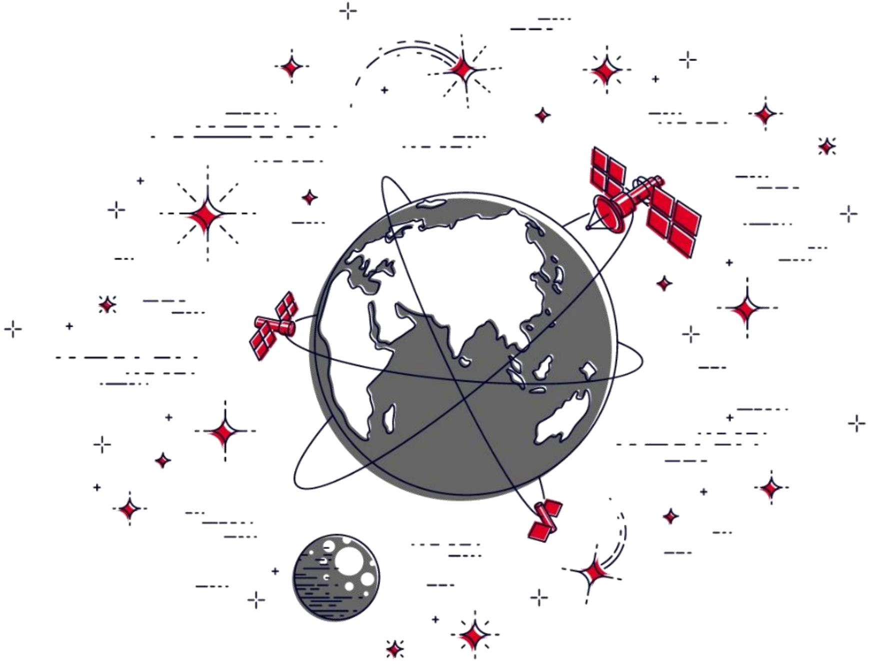
NEXT STEPS

- Enhancement of the simulation environment and integration of the system with other tools.
- Knowledge transfer exploration (sim2real).
- RL sub-fields such as offline learning, meta-learning, and active learning.
- Red Team: Action prioritisation is focused on finding a vulnerable machine and making it their own with maximum privileges.

Questions and Comments

Thank you

GMV, Etamax and ESA



European Space Agency

MEMBER OF FEV GROUP