Final Presentation

INFAST: Intelligent Automated Functional and Security Testing activity

$\ensuremath{\mathbb{C}}$ GMV – All rights reserved

The information contained within this document is considered as "GMV-Limited". The receiver of this information is allowed to use and redistribute the information to current employees of GMV, referring the source of the information; observing legal regulations in intellectual property, personal data protection and other legal requirements where applicable.



European Space Agency





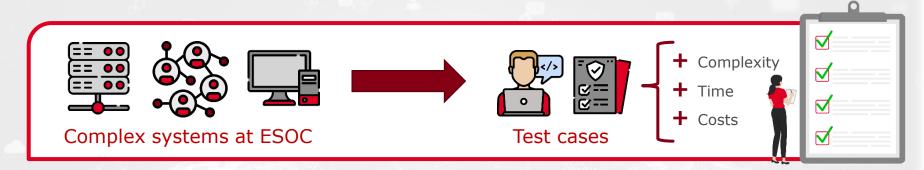
Index

1. Introduction: INFAST project

- 2. Root Cause Analysis for Functional testing
- 3. Action Selection for Automated Pentesting
- 4. Explainable AI
- 5. Conclusions and further work
- 6. Questions and Comments

Introduction

Towards INtelligent automated Functional and Security Testing (INFAST)

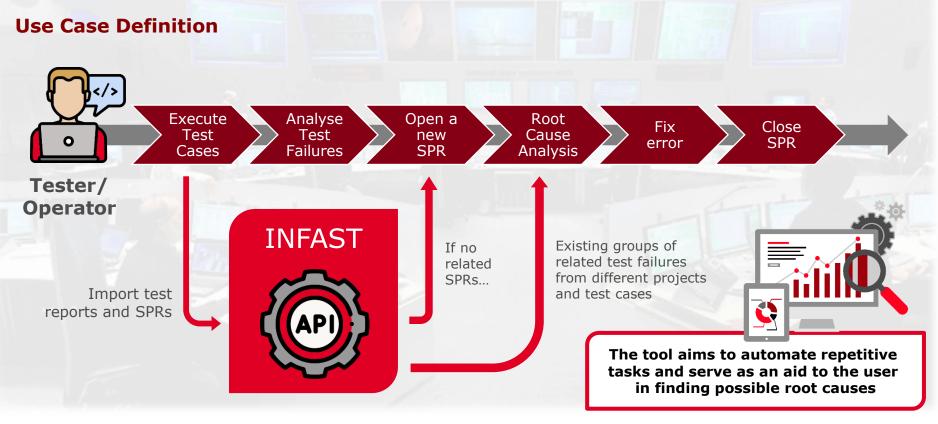


- INFAST is a project funded by ESA aimed to provide solutions enhanced by AI driven algorithms to automate testing tasks.
- ESA funded TDE (Technology Development Element) activity.
- Two Proof-of-Concept use cases -

Root Cause Analysis for Functional testing

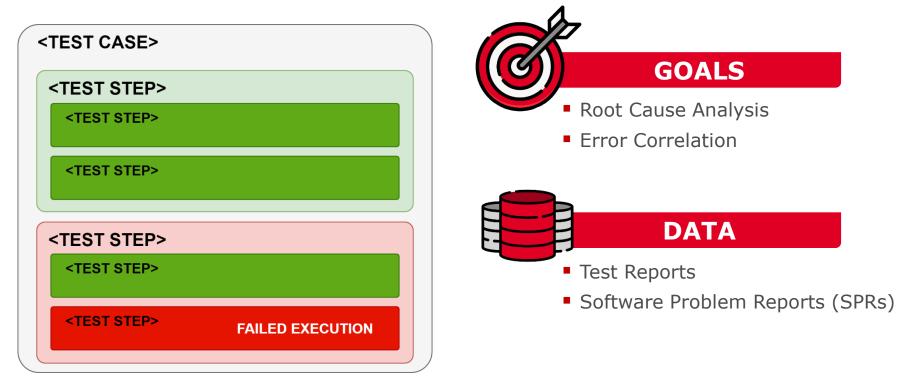
Action Selection for Automated Pentesting







Use Case Definition



esa

gn

Use Case Definition



Natural Language Processing (NLP) approach

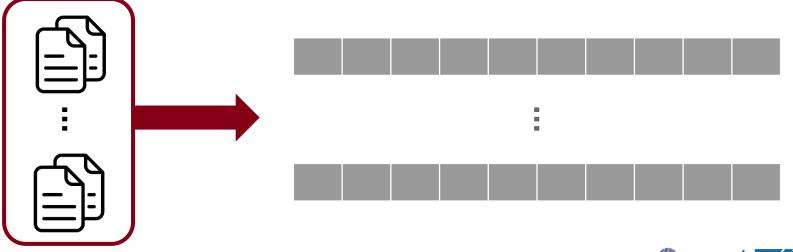
- Extract failures from Test Case.
- Extract text fields from data sources:
 - From test failures: Test step description and execution log.
 - From SPRs: SPR description.
- Obtain relationships based on text similarities.





Proposed approach: Data Encoding

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



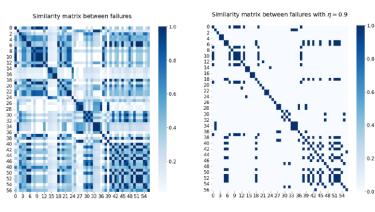


Proposed approach: Similarity

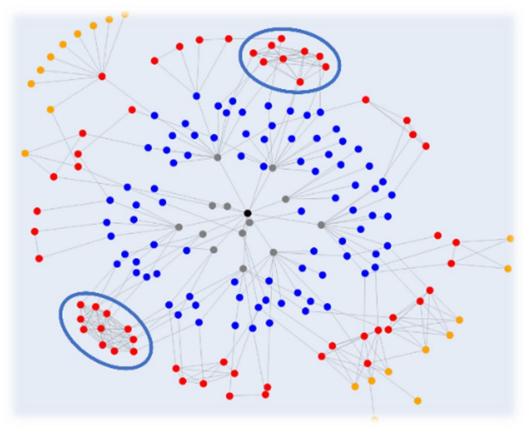
Compute the distance between vectors using the cosine similarity:

$$S_{\mathcal{C}}(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{\|\boldsymbol{x}\| \cdot \|\boldsymbol{y}\|} = \frac{\sum_{i=1}^{n} \boldsymbol{x}_{i} \boldsymbol{y}_{i}}{\sqrt{\sum_{i=1}^{n} (x_{i})^{2}} \sqrt{\sum_{i=1}^{n} (y_{i})^{2}}}$$

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



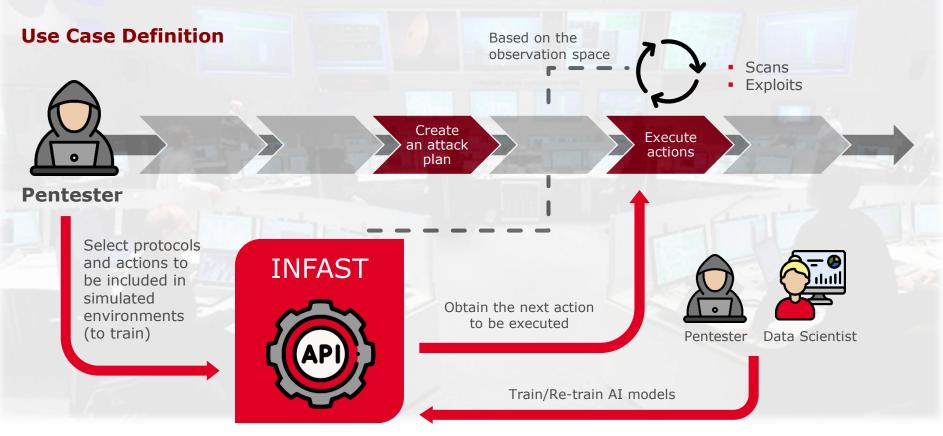




Results

- 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.



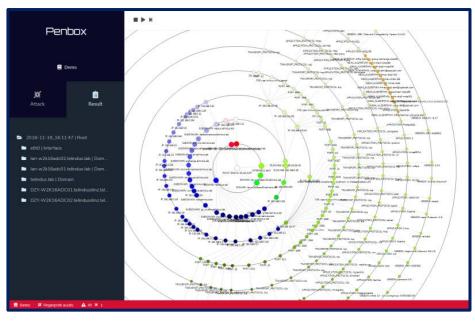


©esa

etamax n

gnv

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.

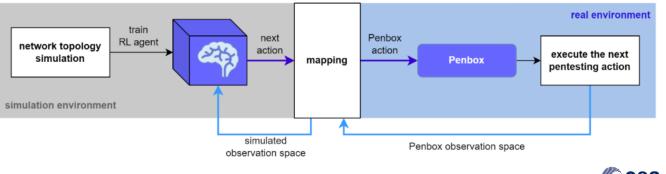


Use Case Definition



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.

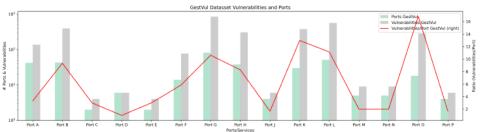


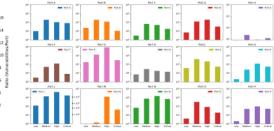


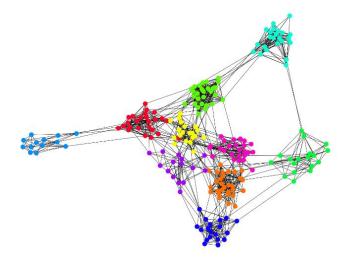


Simulated Environments Generation

- 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









RL Policy: Architectures

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

RL Algorithms: Architectures

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

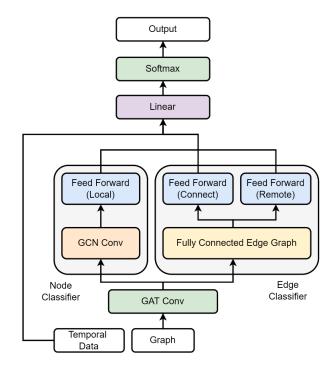


Figure: GNN policy architecture diagram.

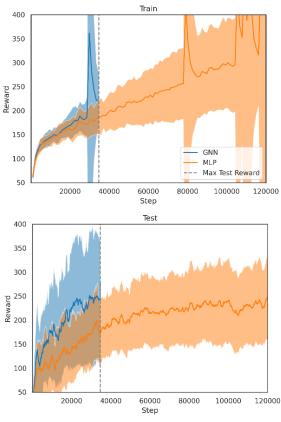


Algorithms/architectures comparison

 Agent achieves converges in both training and testing using DQN.

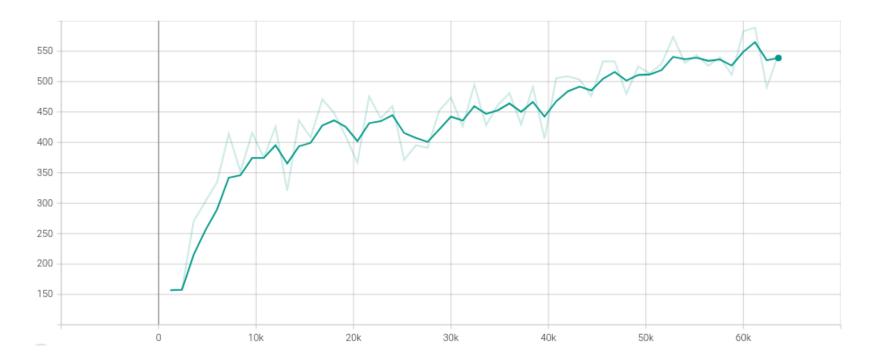
GNNs...

- ...outperforms MLP over the testing step
- Experimentally: Generalizes better achieving maximum test reward
- Theoretically:
 - Permutation equivariance
 - Non-fixed size input & scalability





RL agent training







 \sim

Agent execution in a real Environment (Sim2Real): PenBox

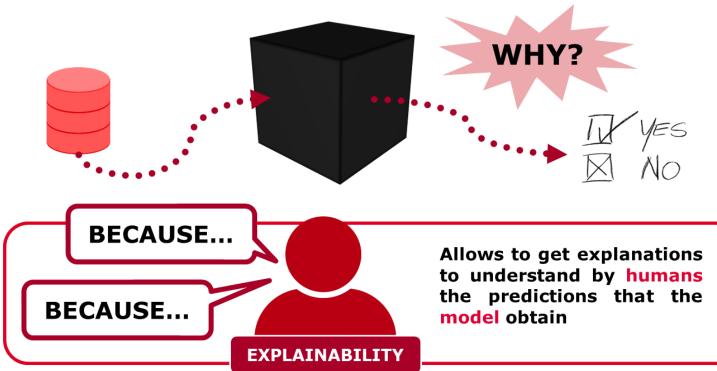
GET /topologies/ Get AI				
POST /topologies/ Create	\sim	Penbox		
GET /topologies/{id} Get By 16	\sim			DNSServiceDiscovery
TELETE /topologies/delete/{id} Deter	\sim			DNSServiceExploitation
PUT /topologies/update/{id} update	~			FTPAnonVulnExploitation FTPCredsCollection
networks	^			FTPServiceDiscovery FTPVulnExploitation
GET /network/ Gelai	\sim	O Infast		HTTPServiceDiscovery MySQLCredsCollection
POST /network/ Create	\sim			MySQLDataEnumeration MySQLHashDump
GET /network/{id} GetBytd	\checkmark	~	<u>~</u>	MySQLSchemaDump MySQLSchemiteDiscovery
DELEYE /network/delete/{id} Dente	\sim	Ø	Ê	PHPVulnExploitation
PUT /network/update/{id} update	~	Attack	Result	PostgreSQLCredsCollection PostgreSQLServiceDiscovery
vulnerabilities	^			PostgreSQLVuInExploitation PrivEscViaPHP
GET /vulnerabilities/ GetAN	\checkmark		* °	Machine Learning PrivescviaPhotographic PrivescviaPhotographic PrivescviaPhotographic PrivescviaTikiwikiSoft
Post /vulnerabilities/ Create	\checkmark	Filter		ScanIDs
GET /vulmerabilities/{id} GetBy M	\checkmark	🗁 Machine Learning Root		SMBServiceDiscovery SMBVulnExploitation SMTPServiceDiscovery SMTPPVulnExploitation SSHServiceOiscovery SSHServiceOiscovery TehnetCredsCollection TehnetServiceDiscovery TikWikiSoftVulnExploitation Use stolen credentials WebsiteDirEnumeration
CELERE /vulnerabilities/delete/{id} Dense	\sim			
Pur /vulnerabilities/update/{id} Update	\checkmark	DNSServiceDiscovery Action		
agent	^	DNSServiceExploitation Action		
FORT /agent/next-action NextAction	\checkmark	🗁 FTPAnonVulnExploitation Action		
POST /agent/load Load	\checkmark	FTPCredsCollection Action		
POST /agent/train Training	\sim	🗁 FTPServiceDiscovery Activity		
POST /agent/inference Getinteence	\checkmark	FTPVulnExploitation Action		WebsiteVulnScan
GET /agent/ GerAll	\sim			

topologies



Explainable Artificial Intelligence

Context







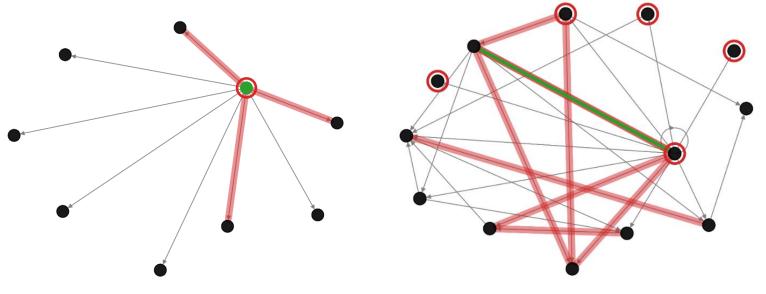
gn

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

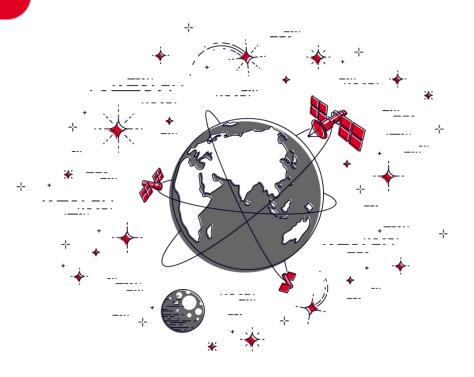




gmv.com

Thank you

GMV, Etamax and ESA





© GMV – All rights reserved

GMV-INFAST-FP, 07/03/2023