Date 29/09/2023

Challenge 01 "Automation of cybersecurity verification for cyber physical systems"

Challenge 01 Working Group Meeting

Philippe Massonet, Coordinateur Scientifique CETIC
Guillaume NGuyen, UNamur
Martin Vivian, UCLouvain
Denis Darquennes, CETIC

digital wallonia 4.cyberwal

Wallonie recherche SPW

https://cyberwal.be
https://cyberexcellence.be

Agenda

15:00-15:20	Improvements for stateful fuzzing	Martin Vivian (UCLouvain)
15:20-15:40	Identification of Cyber Physical System (CPS) & Orchestration of fuzzing testing	Guillaume Nguyen (Unamur)
15:40-16:00	Automated cybersecurity testing with genetic algorithms	Denis Darquennes, Philippe Massonet (CETIC)

Challenge 01 "Automation of cybersecurity verification for cyber physical systems"

- **Summary of the Challenge:**
  - Penetration testing: still a very manual process, requires cybersecurity experts

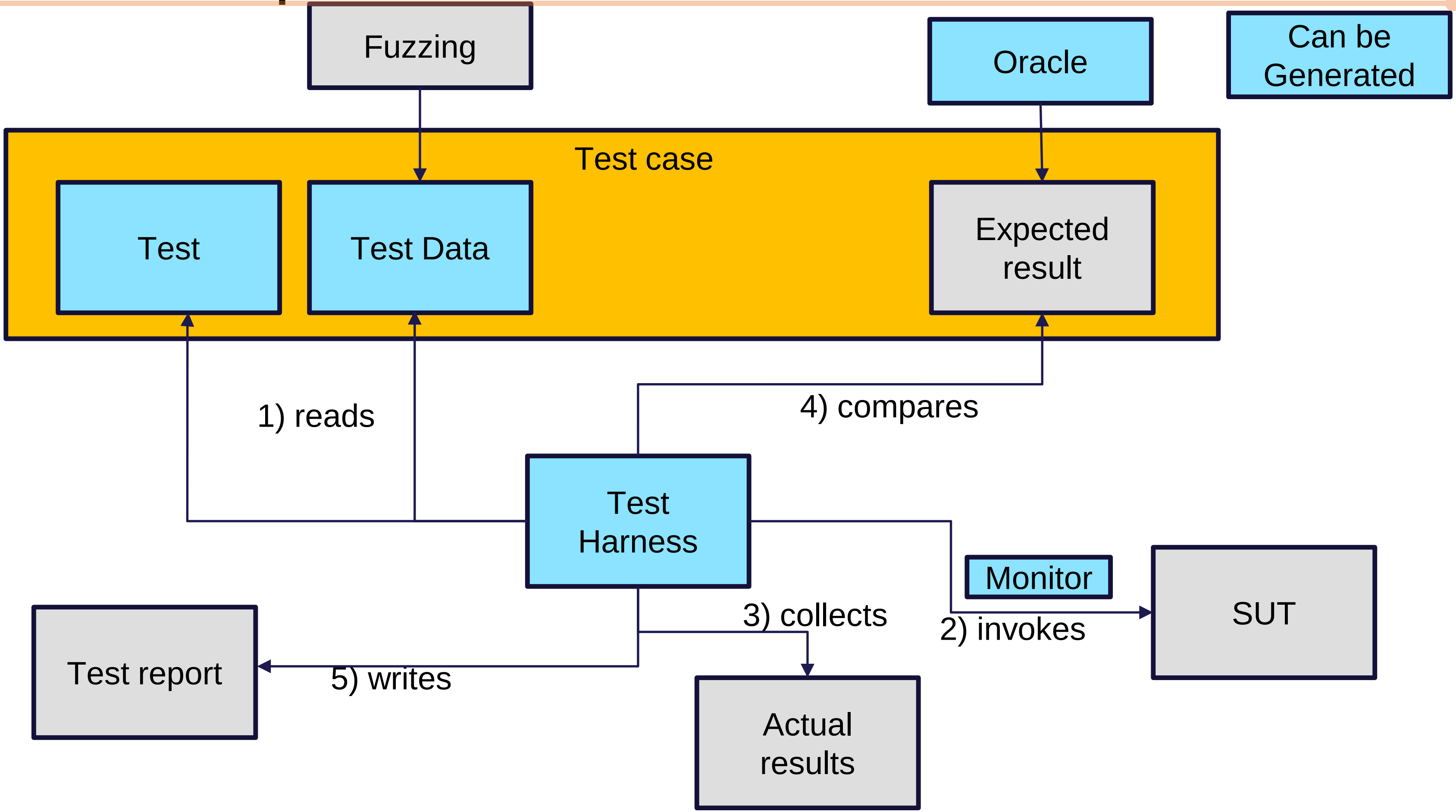
  - Ambition: automate (partly) the creation of penetration tests to make penetration tests more accessible to companies (SMEs, large companies)
- **Research Challenges**:
  - Automatic generation of functional cybersecurity tests (security architecture), use of different generation techniques (to compare) for penetration tests:

    - Fuzzing techniques,
    - Generation of tests by genetic mutation

    - Generating tests from models

    ...
  - Partial automation in the form of assistance with the creation process and definition of penetration tests.

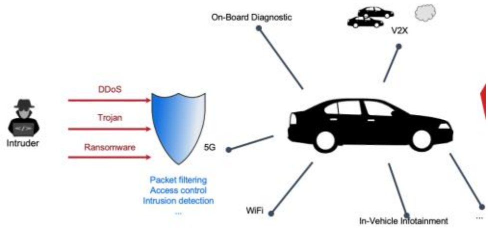
“Test Harness pattern” - Generation

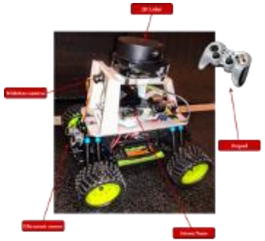


Défi 01 : Automatisation de la vérification cybersécurité de systèmes cyber physiques - consultation – Problèmes de recherche

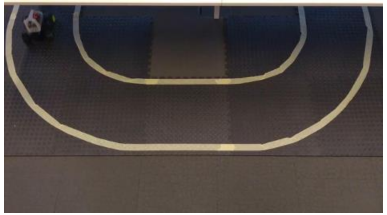
- Problèmes de recherche
  - UNamur : fuzzing guidé par des algorithmes génétiques (Prof. Xavier Devroey, 1 chercheur)
  - UCLouvain : fuzzing – découverte de protocoles de communication par apprentissage, analyse de malware (Prof. Axel Legay, 2 chercheurs)
  - CETIC : génération de jeux de tests, proposition d’une étude de cas (Philippe Massonet, 2 chercheurs pour 1ETP)

- Expérimentation dans la factory
  - Déploiement d’un système à tester dans une sandbox de la factory
  - Etude de cas :
    - véhicule connecté (V2X)+ centre de gestion de traffic (Cloud)
    - Introduction de vulnérabilités
  - Challenge : tests générés découvrent-ils les vulnérabilités/malware





Rover - Donkey Car

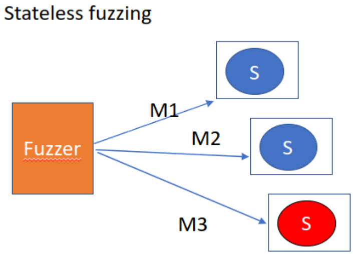


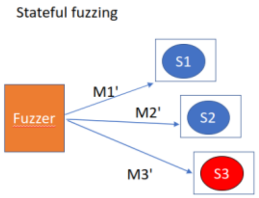
Improvements for stateful fuzzing

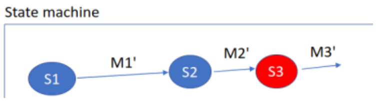
Martin Vivian, UCLouvain

Reminder

Fuzzing on State Machine

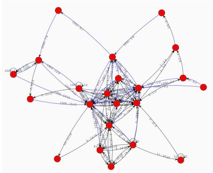






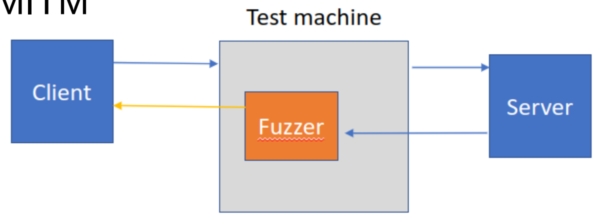
Each message have their grammar
Order of the message in this example, we must send M1' before M2' to reach S3

State machine

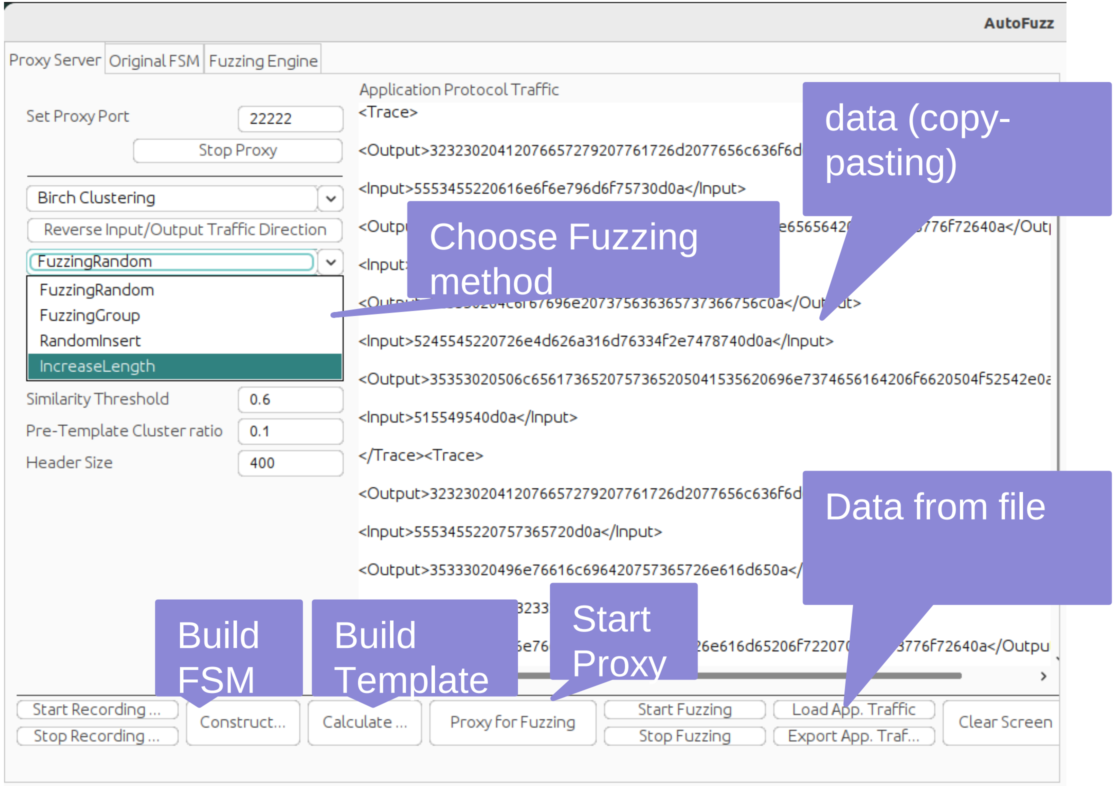


Template of message
for each state transition

MITM



Tool Introduction



Set Proxy Port: 22222

FuzzingRandom

Similarity Threshold: 0.6

Pre-Template Cluster ratio: 0.1

Header Size: 400

```
<Trace>
<Output>32323020412076657279207761726d2077656c636f6d
<Input>5553455220616e6f6e796d6f75730d0a</Input>
<Outp                                          e65656420                 5776f72640a</Out
<Input
<Out                 67696e2073756363657373666756c0a</Out    ut>
<Input>5245545220726e4d626a316d76334f2e7478740d0a</Input>
<Output>35353020506c65617365207573652050415356206e737465616420f6620504f52542e0a
<Input>515549540d0a</Input>
</Trace><Trace>
<Output>32323020412076657279207761726d2077656c636f6d
<Input>55534552207573657220d0a</Input>
<Output>3533020496e76616c69642075736572616d650a</
              3233
             5e76                     26e616d65206f72207              3776f72640a</Outpu
```

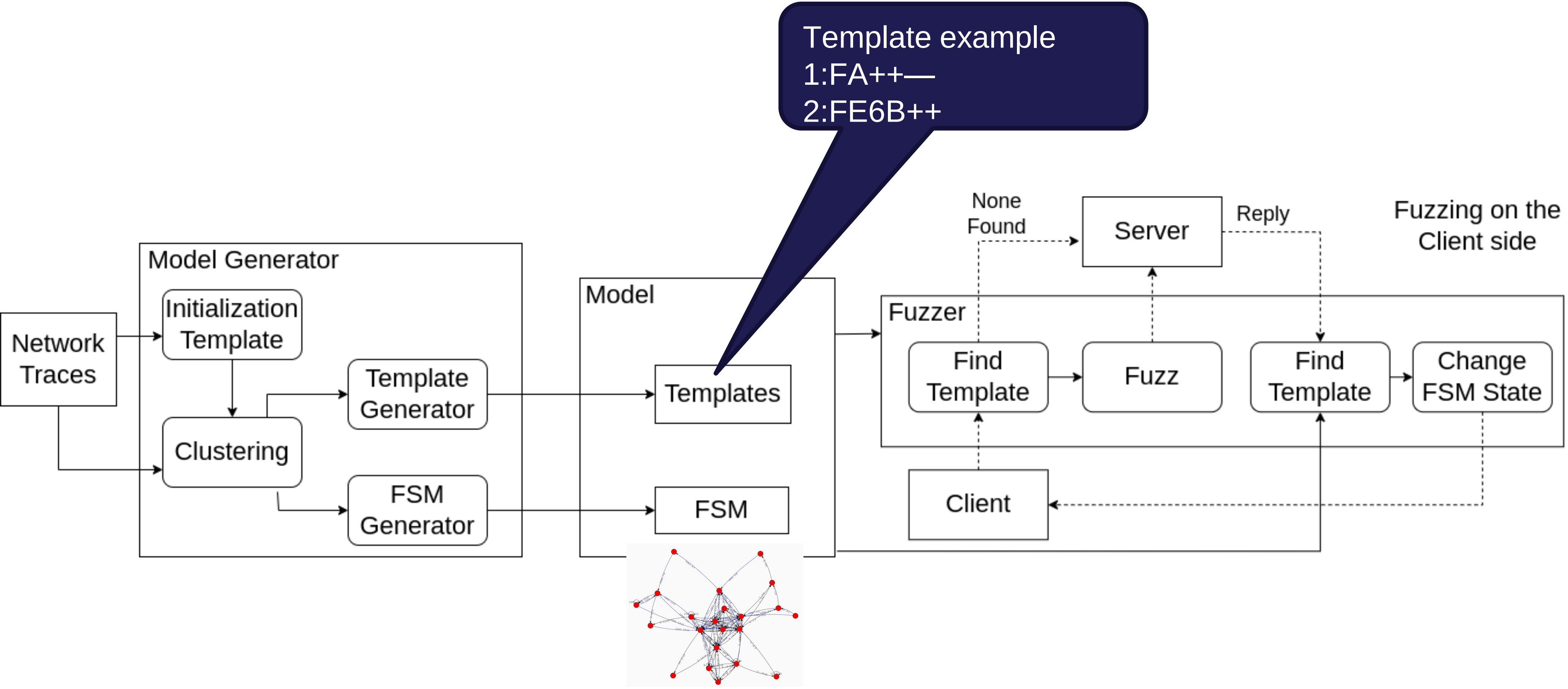
Data in XML format

<Trace></Trace> : session
<Input></Input> : data
<Ourput></Output>

Initially start to improve Autofuzz : https://sourceforge.net/projects/autofuzz/

Finally keep UI and modify the rest

Process of the Tool



Template example
1:FA++—
2:FE6B++

Network Traces

Model Generator

Initialization Template

Clustering

Template Generator

FSM Generator

Model

Templates

FSM

None Found

Server

Reply

Fuzzing on the Client side

Fuzzer

Find Template

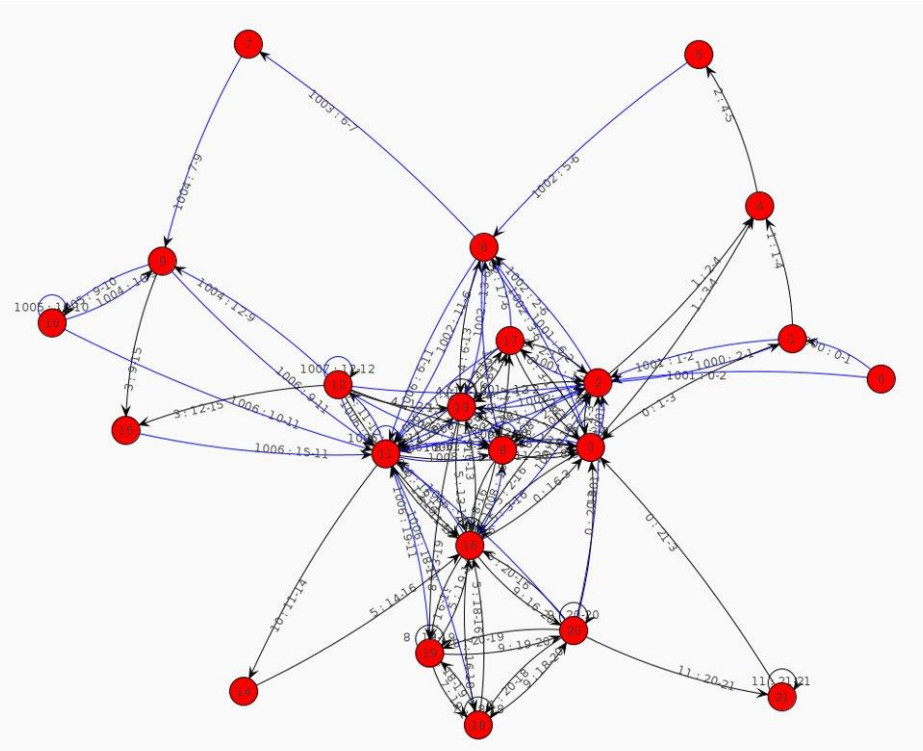
Fuzz

Find Template

Change FSM State

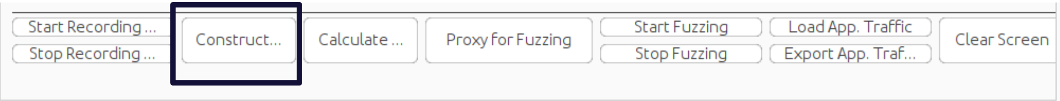
Client

Build State Machine



Integrated tool for FSM

FlexFringe : https://github.com/tudelft-cda-lab/FlexFringe



Clustering for State Machine

Gather similar networking messages from the data to build a fsm based on this cluster

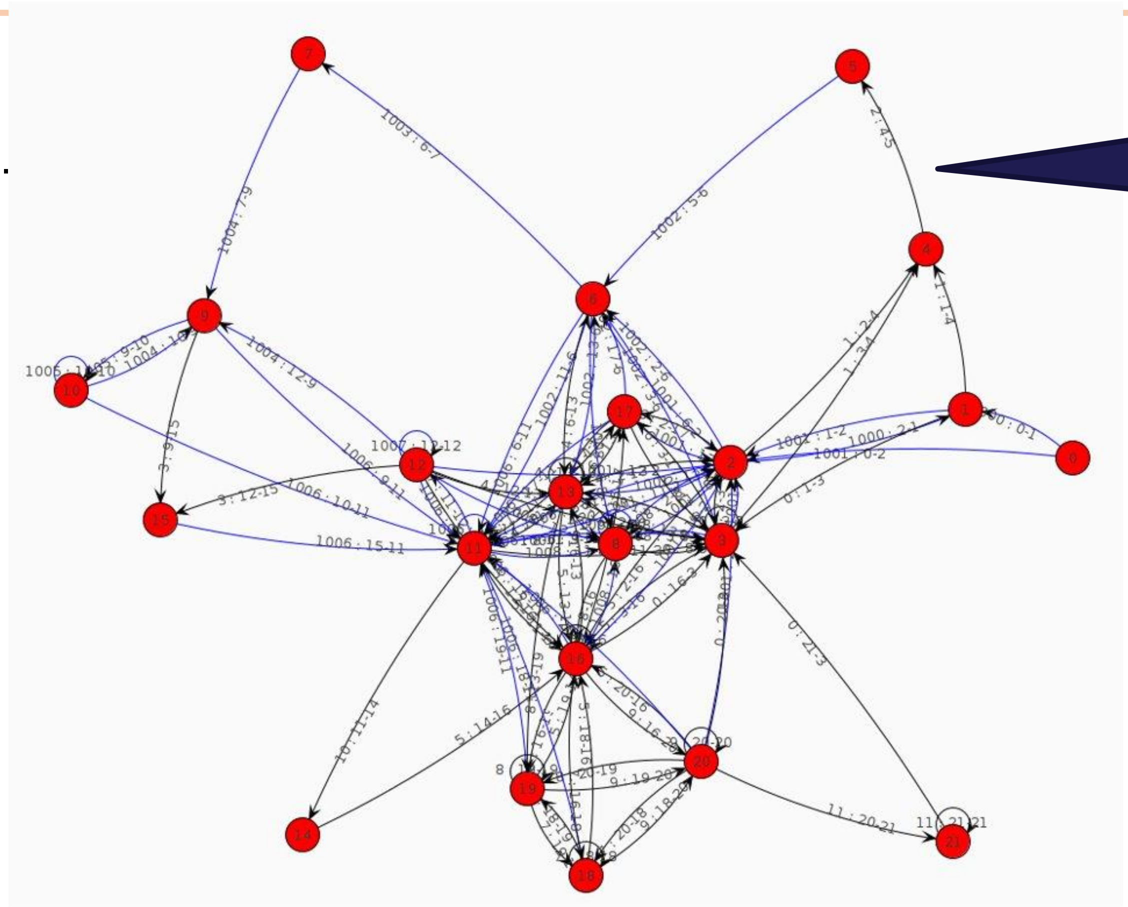
1) Pretemplate :
Find highest variable part (like crc, session id...) and don’t take in account for clustering
Example : FE+++++++A——-+++—- => “+” indicate high variablility

2) Header separation (clustering on header) :
Choose an header length for the clustering
Example : header = 4 for frame “AABBCCDD” => header part is AABB
Create separate cluster for size below
Create a separate cluster for unclusterized data

3) Algorithm used
Hiearchical Clustering algorithm :
(BirchLeaf clustering) : https://github.com/sbobek/smiling/blob/master/demo/src/main/java/smile/demo/vq/BIR

Link between State Machine and cluster

..



- We are at the state 4
- We receive an ouptut message that match the cluster ID 2 then we go to the state 5

Cluster Id 2 should be represented by the template : AF++-C—++
And the message received should be AFEBFCEEAA

If output are fuzz then the message will be fuzzed following the corresponding template

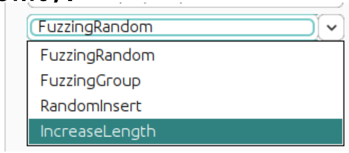
Template and Fuzzing Strategy

Template :

- Since the fuzzer is in MitM we can modify the messages by following a template
- We need before to calculate the template for each cluster
- Distinction between constant value, strong and weak variable with Xi² (distribution for each position in the frame).

Strategy :

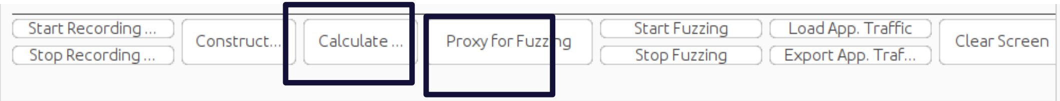
- Don’t fuzz constant, less fuzzing for highest variable and high fuzzing for weak variable. Template example : FE++++—— (red no fuzzing, yellow low probability to be fuzzed and green higher probability.
- Fuzzing Function :



- Possibility to not fuzz all frame, select states to fuzz.

Information :

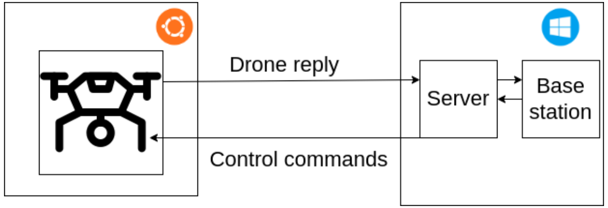
Distinction beween constant value, strong ar



Application to Industrial drone Case Study

Industrial protocol caracterised by

- CRC
- Session ID
- DateTime
- Telemetric, logs packets
- Header
- Black-box
- No always tuple input-output
- No text-interpretable protocol



Results on the case study

- Relevant State machine
- Relevant template to identify cluster and reusable for fuzzing
- Tools is enough good to create separate state for the drone commands
  - For example each time that we start the motors we go to the state 9.
  - It gives the possibility to only fuzz specific control command without modifying telemetries packet and get the good fuzzing template .
- For the moment on the case studies with this we can create protocols errors but not really crash or vulnerability.

Application to the Rover Case Study and demonstration plans

- We could directly test our tools on the Rover case study by following the same strategy of the drone.
- Some parameters should be change for example : header size, cluster number, size of dataset, probability to fuzz...
- The difficulty should be to place the tools between the rover. It may be necessary to integrate the possibility of injecting data via a MitM attack.

Next Steps and progress in maturity of results

- Test on more case studies
- Fix last issues : the tool sometimes seems to miss to fuzz some packets, the fsm build use too many memory => limitation for the dataset size.
- Find an heuristics to know in advance the number of clusters
- Possibility to correct the model during the fuzzing phases

Test-based classification framework for CPS

Guillaume Nguyen, UNamur

Table of contents

1. Context

2. Survey

3. Challenges

4. Conclusion

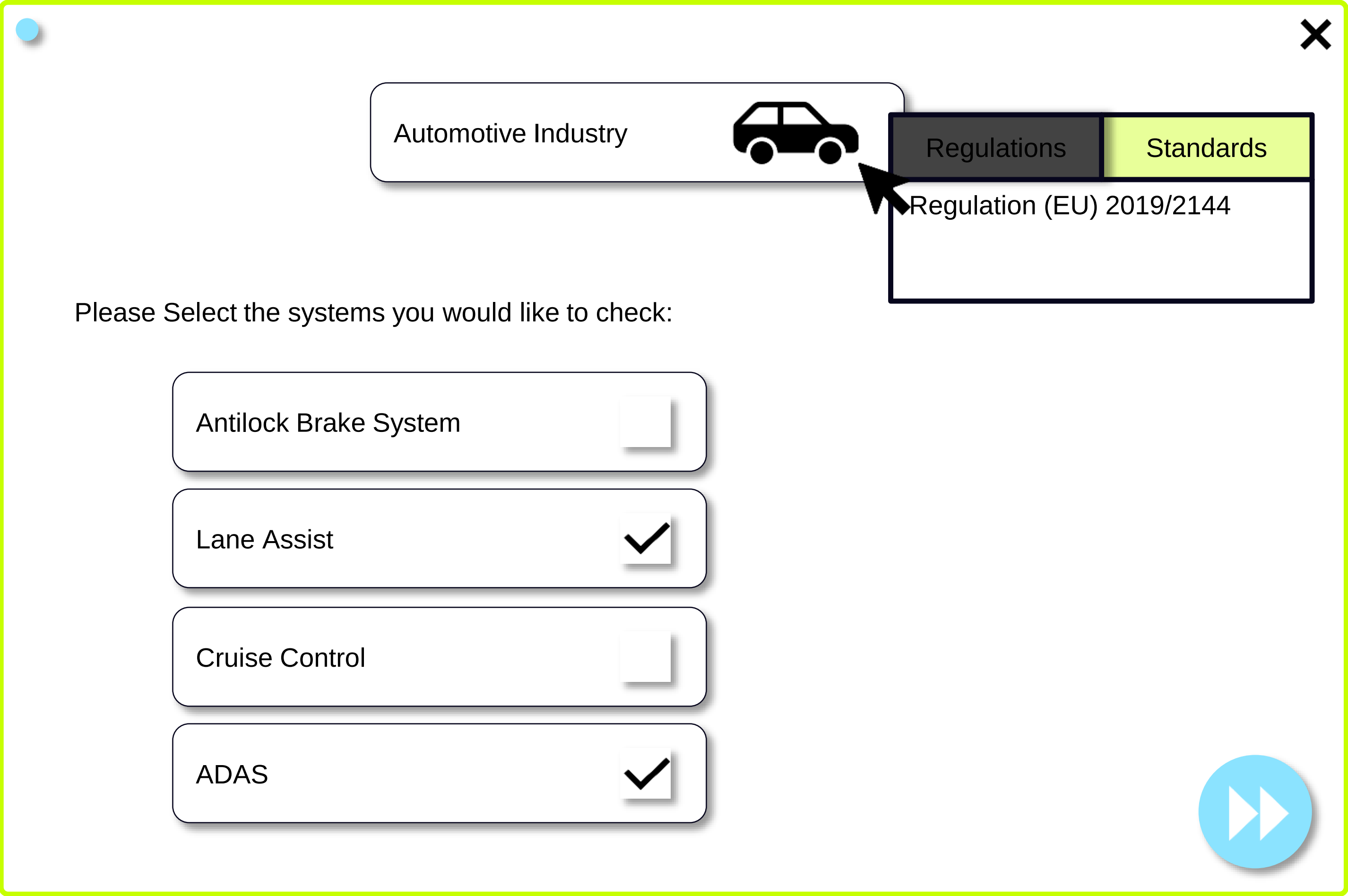
Context

Ambient Intelligence

Proprietary Technologies

Versatility

Fuzzing



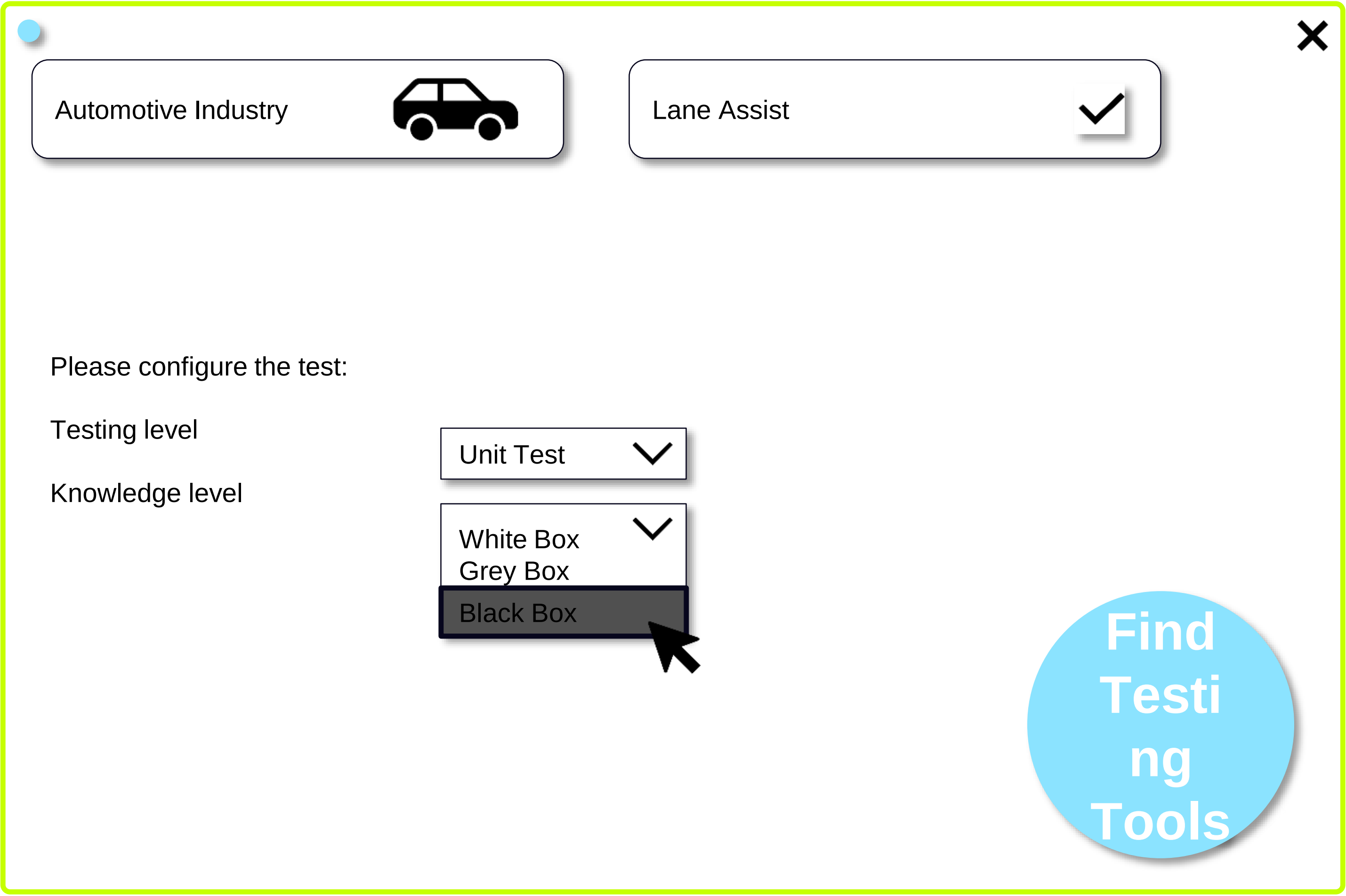
Automotive Industry

Regulations | Standards

Regulation (EU) 2019/2144

Please Select the systems you would like to check:

- Antilock Brake System □
- Lane Assist ☑
- Cruise Control □
- ADAS ☑



Automotive Industry

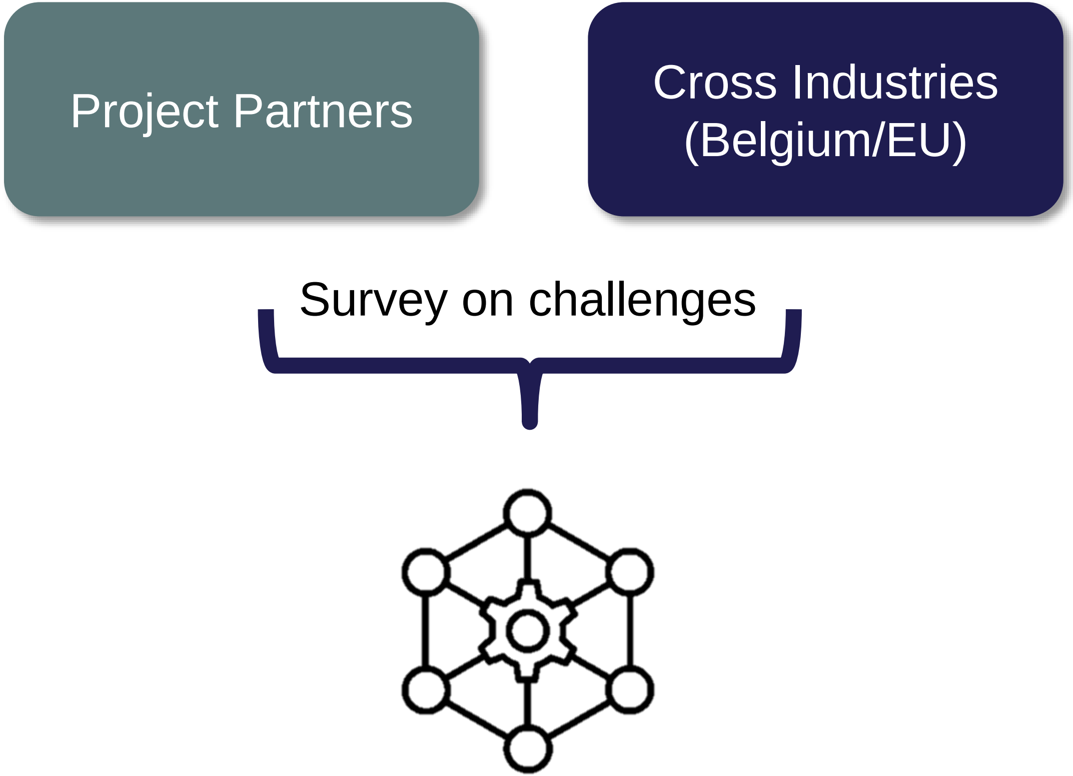
Lane Assist ✓

Please configure the test:

Testing level: Unit Test

Knowledge level:
White Box
Grey Box
Black Box

Survey



Project Partners

Cross Industries (Belgium/EU)

Survey on challenges

Since May, only 5 usable responses to the questionnaire

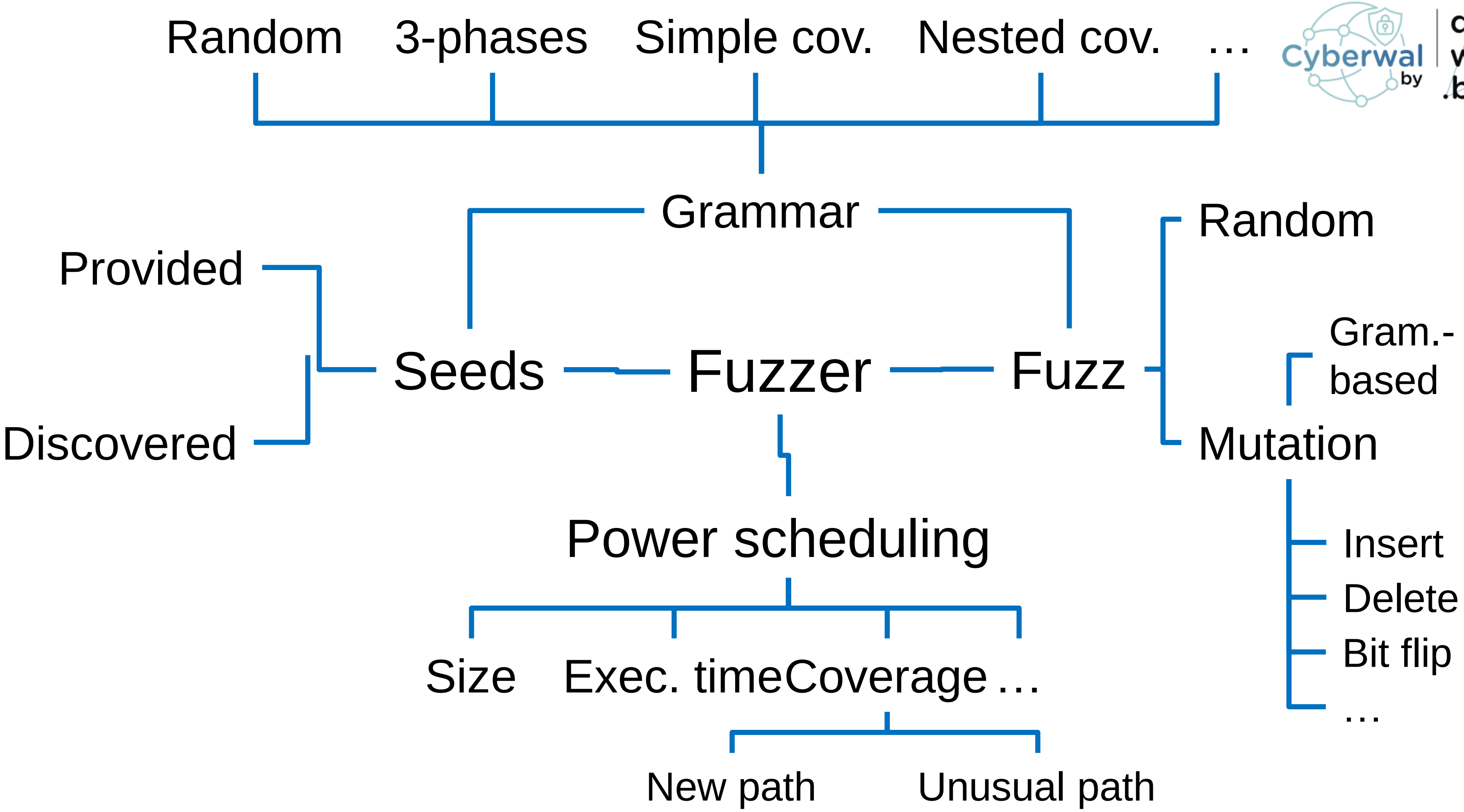
Challenges

- How can we ensure overall quality through time?
    - Regulations adaptations
    - Internal processes changes
    - Standards developments
    - Industry needs évolutions

We don’t want to make new products every 30 years!

Fuzzing as a one-go technique

- We already identified that CPS were developed and operated using various technologies, constructors, protocols, etc.
- Fuzzing allows for various levels of agnosticism. It has capabilities ranging from brute–force to very specific surgical types of attacks.
- Multiple tools already exist it’s all about finding the good one.



Random 3-phases Simple cov. Nested cov. …

Grammar

Provided

Seeds Fuzzer Fuzz

Discovered

Random

Gram.-based

Mutation

Power scheduling

Insert

Delete

Bit flip

…

Size Exec. time Coverage …

New path Unusual path

Conclusion

- We need more responses for a more qualitative analyses of the test challenges when it comes to CPS
- We will perform multiple case studies to spot the differences accross industries
- Dynamic testing and evolution will benefit industries, customers and states
- We will identify the most suitable fuzzing tests and tools

digital wallonia 4 cyberwal

Automated cybersecurity testing with genetic algorithms

Denis Darquennes, Philippe Massonet

How can genetic algorithms be used for Cybersecurity testing ?

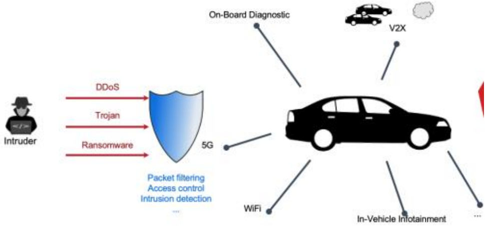
Cybersecurity test generation methods and tools based on **genetic algorithms**

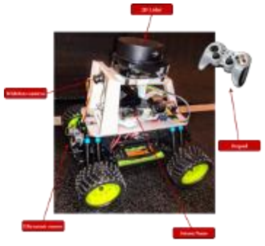
Tools (White box):
- Pynguin (Python)
- EvoSuite (Java)
- Mull (C/C++)

Not cybersecurity specific

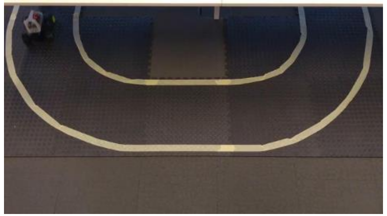
Cybersecurity architecture testing (CIA) ?

Penetration testing, Vulnerability analysis ?



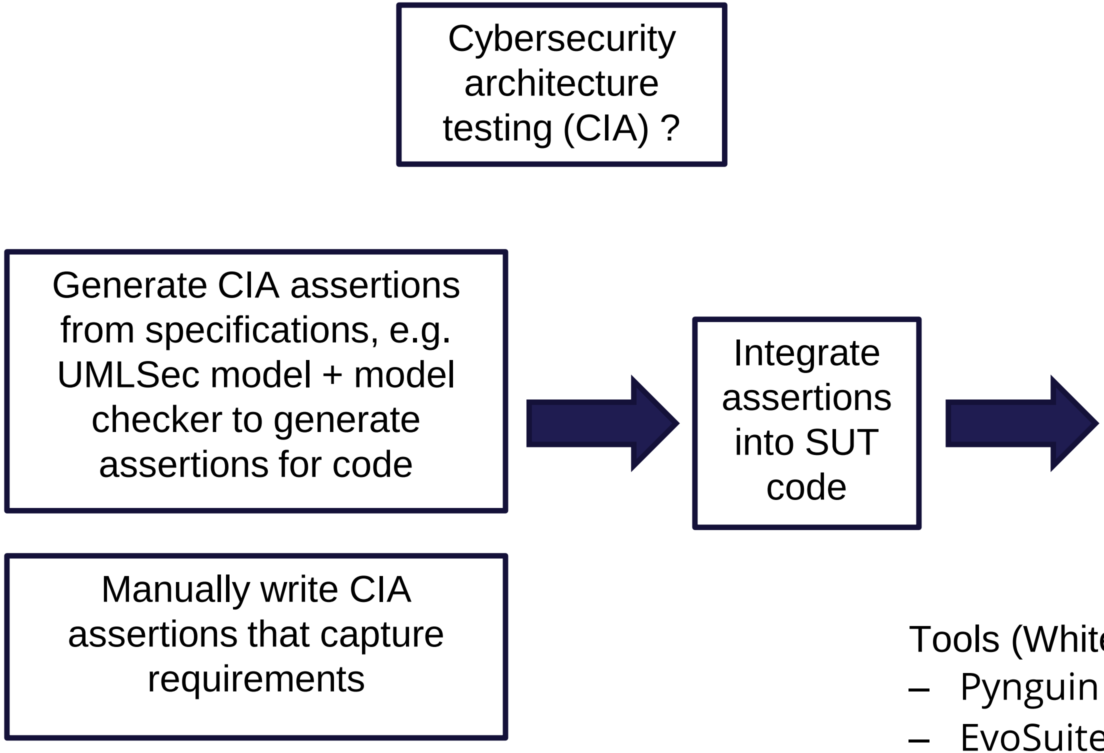


Rover - Donkey Car



Platooning case Study with rovers (CPS)

Methodology for Cybersecurity architecture testing (CIA)

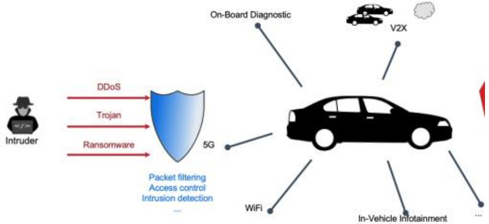


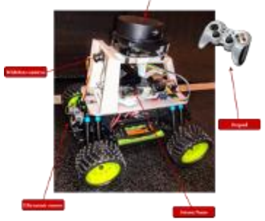
Cybersecurity architecture testing (CIA) ?

Generate CIA assertions from specifications, e.g. UMLSec model + model checker to generate assertions for code

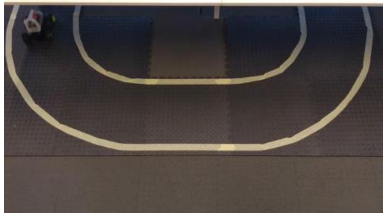
Manually write CIA assertions that capture requirements

Integrate assertions into SUT code





Rover - Donkey Car



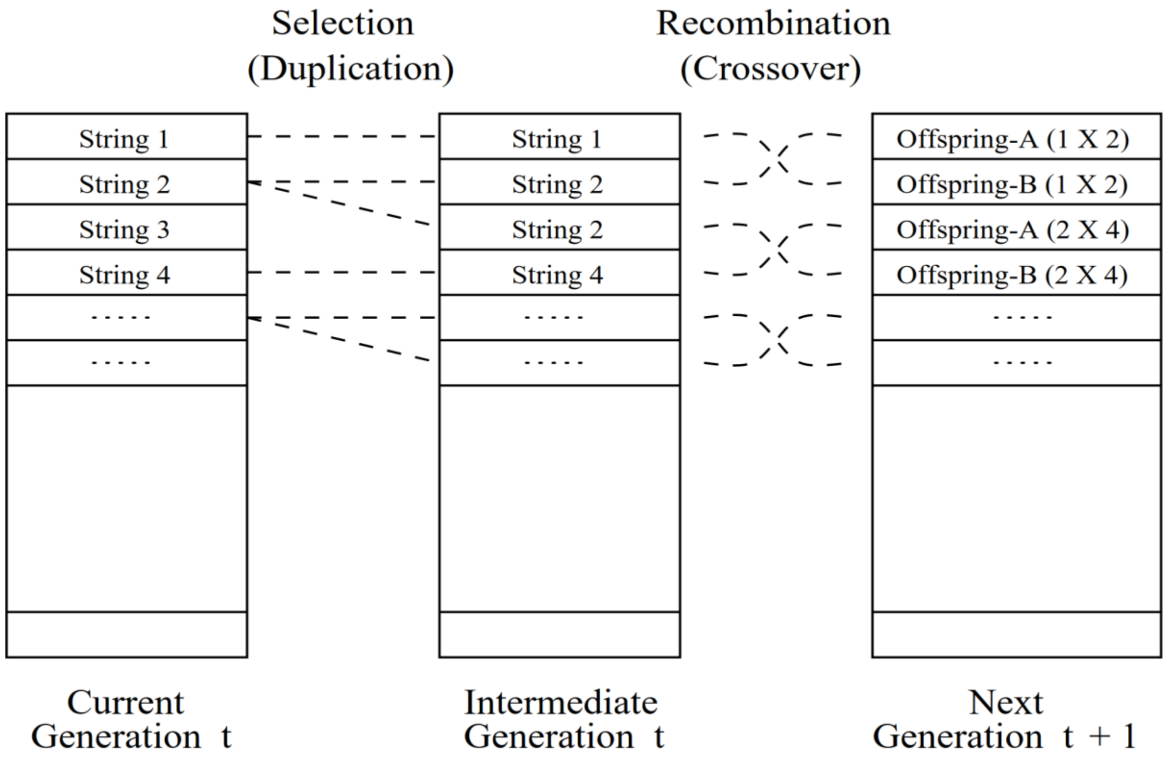
Tools (White box):

- Pynguin (Python)
- EvoSuite (Java)
- Mull (C/C++)

Platooning case Study with rovers (CPS)

Why genetic algorithms for software cybersecurity testing

From : “A Genetic Algorithm Tutorial”, D. Whitley, Springer 1994



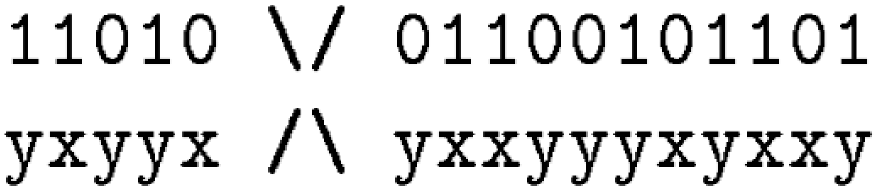
Selection (Duplication)

Recombination (Crossover)

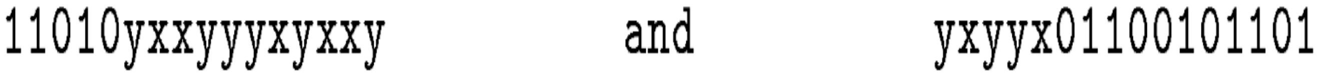
Current Generation t	Intermediate Generation t	Next Generation t + 1
String 1	String 1	Offspring-A (1 X 2)
String 2	String 2	Offspring-B (1 X 2)
String 3	String 2	Offspring-A (2 X 4)
String 4	String 4	Offspring-B (2 X 4)
. . . . .	. . . . .	. . . . .
. . . . .	. . . . .	. . . . .

Why genetic algorithms for software cybersecurity testing (1/2)

Example of recombination



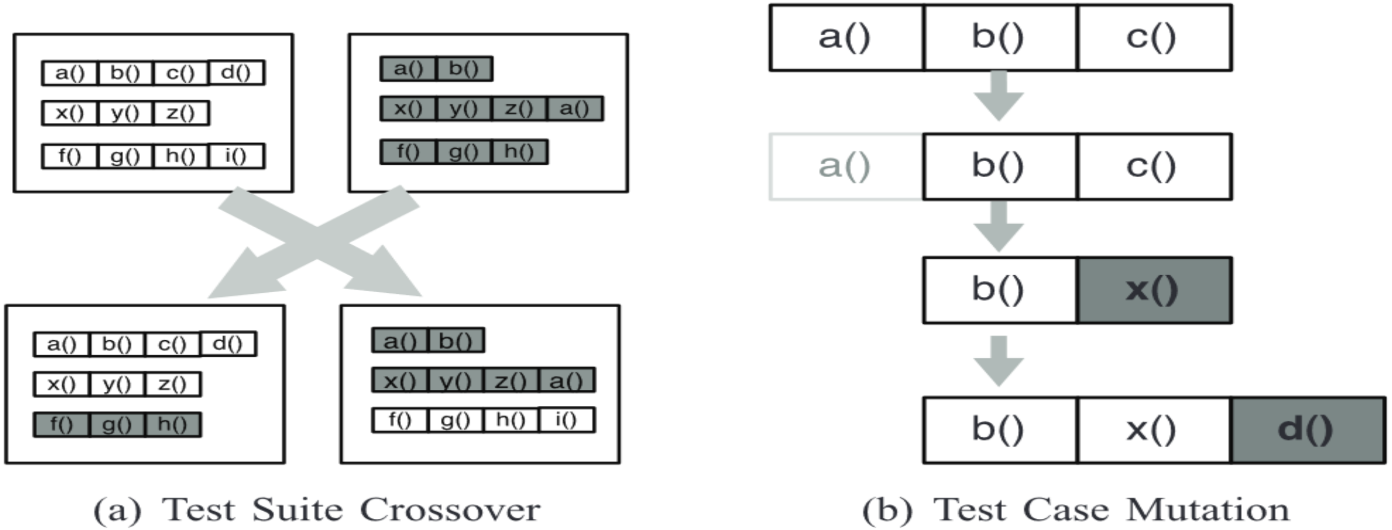
```
11010 \/ 01100101101
yxyyx /\ yxxyyyxyxxy
```



11010yxxyyyxyxxy and yxyyx01100101101

EvoSuite (Java) : Test Suite Generation

- https://www.evosuite.org/evosuite/
- Test suite generation
- The individuals of population are sets of test cases (test suites)
- Crossover is based on exchanging test cases
- Mutation adds/modifies tests to suites, and adds/removes/changes statements within tests
- Fitness function estimates how close a test suite is to covering all branches of a program



(a) Test Suite Crossover (b) Test Case Mutation

- (From “Whole Test Suite Generation”, G. Fraser, A. Arcuri, IEEE Trans. on Sftw Eng, 2013)

EvoSuite (Example)

```
public void test3()  throws Throwable  {
    TestReentrantLock testReentrantLock0 = new TestReentrantLock();
    testReentrantLock0.methodeB(1626);
    testReentrantLock0.methodeB(1626);
    String[] stringArray0 = new String[5];
    stringArray0[0] = "jZMv]:•2V0B{0a/#,,O";
    stringArray0[1] = "";
    stringArray0[2] = "bF\"^2;cOM^MXVz-H•";
    stringArray0[3] = "/=V5Qu(=hv-Q!d<!U";
    stringArray0[4] = ":<s3K4p{sE";
    TestReentrantLock.main(stringArray0);
    testReentrantLock0.methodeB(1626);
    testReentrantLock0.methodeB(1);
    testReentrantLock0.methodeA(1626);
    testReentrantLock0.methodeB(720);
    testReentrantLock0.methodeA(1626);
    testReentrantLock0.methodeA((-2314));
    testReentrantLock0.methodeB(42);
    testReentrantLock0.methodeA((-1));
    testReentrantLock0.methodeB(0);
    testReentrantLock0.methodeB(720);
    testReentrantLock0.methodeA(0);
    testReentrantLock0.methodeB(1626);
    testReentrantLock0.methodeA((-1));
```

Pynguin (Python) (cont)

- https://pynguin.readthedocs.io/en/latest/index.html

- Generation of regression assertions within its generated test cases based on the values observed during test-case execution

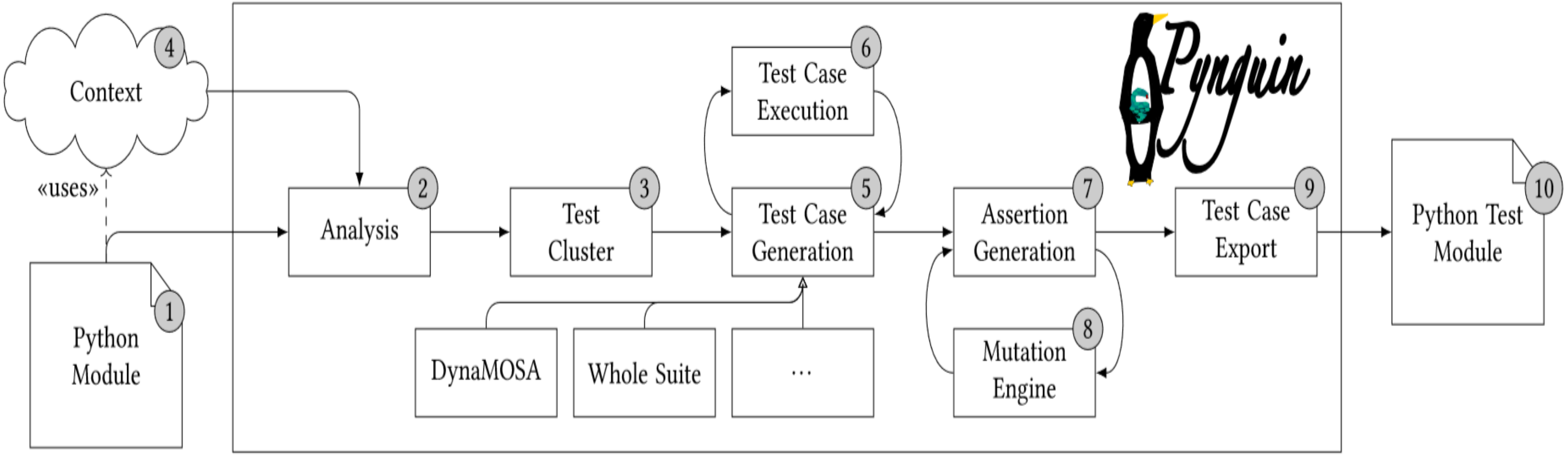
- Need to analyse code, assertable types : enum values, builtin types (int, str, bytes, bool and None), builtin collection types (tuple, list, dict and set) + assertion generation on raised exceptions

- Observed properties: return values of functions, method invocations

Pynguin (Python)

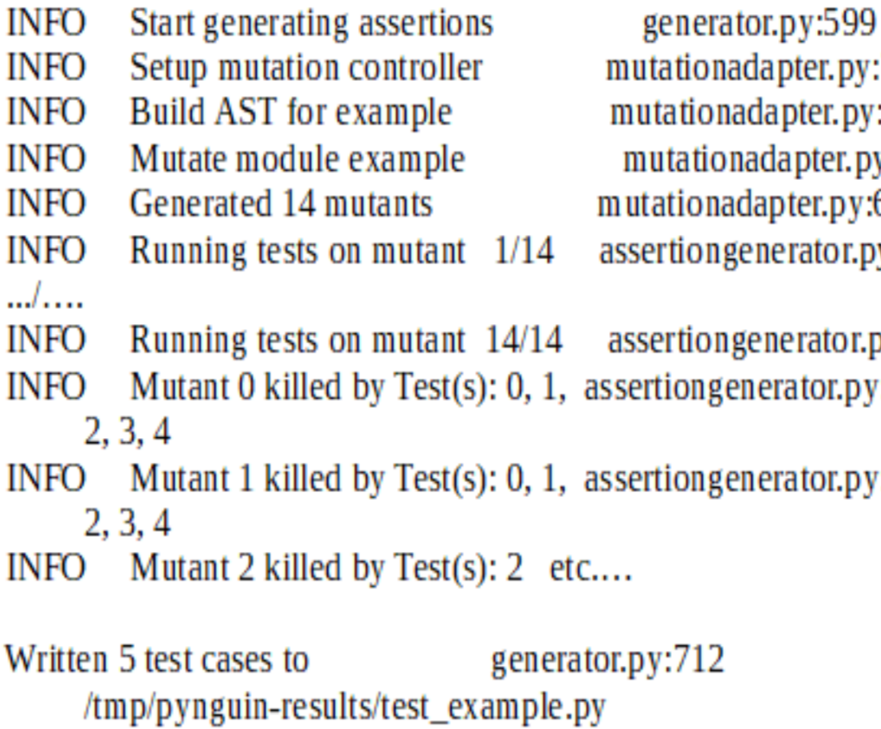
https://pynguin.readthedocs.io/en/latest/index.html

(from “Pynguin: Automated Unit Test Generation for Python”, S. Lukasczyk, G. Fraser



Pynguin (Python)

```
def triangle(x: int, y: int, z: int) -> str:
  if x == y == z:

    return "Equilateral triangle"

  elif x == y or y == z or x == z:

    return "Isosceles triangle"

  else:

    return "Scalene triangle"

import example as module0

    def test_case_0():
        complex_0 = -298.067 + 2058j
        bool_0 = True
        str_0 = module_0.triangle(complex_0, complex_0, bool_0)
        assert str_0 == "Isosceles triangle"

    def test_case_1():
        str_0 = 'M8M-3]Q.(UDhzK`5Bc"'
```

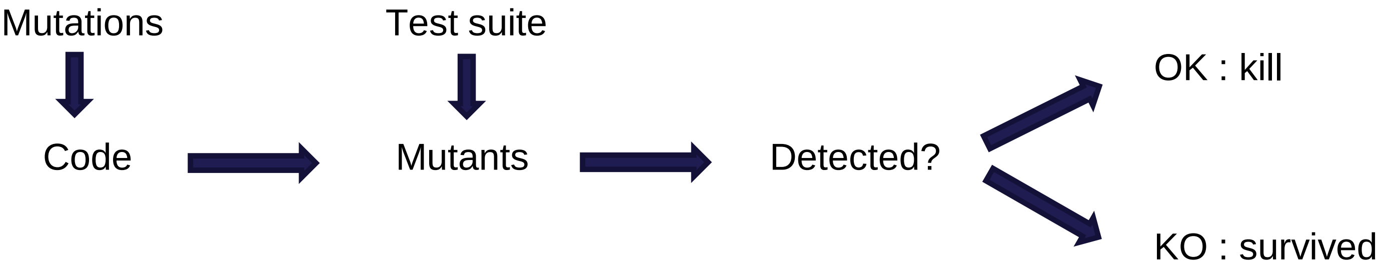


```
INFO   Start generating assertions             generator.py:599
INFO   Setup mutation controller             mutationadapter.py:
INFO   Build AST for example                  mutationadapter.py:
INFO   Mutate module example                   mutationadapter.py
INFO   Generated 14 mutants                  mutationadapter.py:6
INFO   Running tests on mutant  1/14      assertiongenerator.py
.../....
INFO   Running tests on mutant  14/14      assertiongenerator.p
INFO   Mutant 0 killed by Test(s): 0, 1,  assertiongenerator.py
       2, 3, 4
INFO   Mutant 1 killed by Test(s): 0, 1,  assertiongenerator.py
       2, 3, 4
INFO   Mutant 2 killed by Test(s): 2   etc....

Written 5 test cases to               generator.py:712
       /tmp/pynguin-results/test_example.py
```

Mull (C/C++)

https://github.com/mull-project/mull

Evaluate and **improve** quality of software existing tests (does not generate new tests)



Mutations

Test suite

Code

Mutants

Detected?

OK : kill

KO : survived

Maybe useful to complement a tool that generates tests

Application to the Rover Case Study and demonstration plans

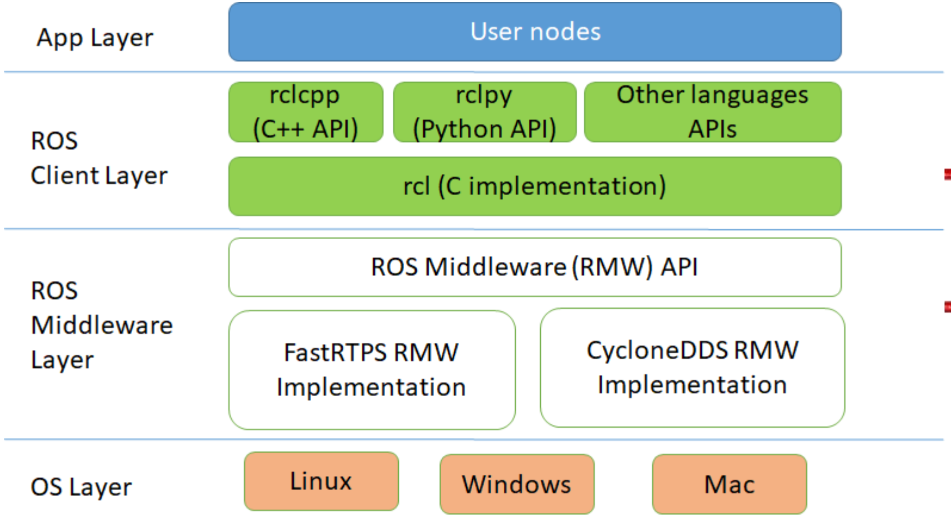
EvoSuite (Java)

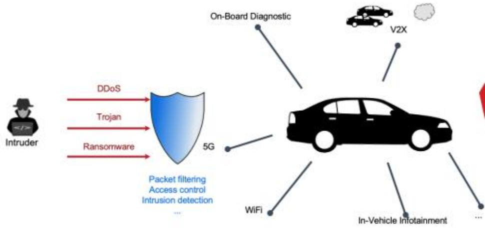
Pynguin (Python)

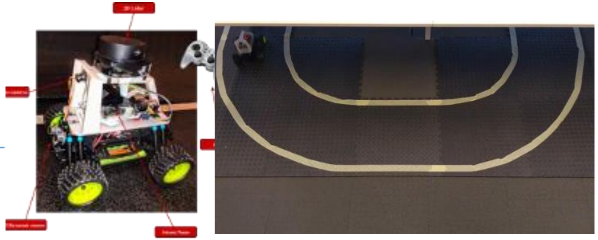
Mull (C/C++)

...

Traffic Supervision (Java)







Rover - Donkey Car

CPS case Study

Next Steps and progress in maturity of results

- Cybersecurity architecture testing (CIA)
  - Place testing tools in a Kubernetes Container
  - Apply test generation tools to the rover platoon case study
  - Try other genetic algorithm based test case generation tools, e.g. for C/C++
  - Try other tools that genetic algorithms

- Penetration testing, Vulnerability analysis ?
  - tests can reveal faults that can become vulnerabilities?

Planning réunion de groupe de travail par Défi

Date	Description
23/01/2023	First meeting of the working group
29/09/2023	Présentation des research results and discussion on demonstrators
*/01/2024	Présentation of démonstrateurs
*/06/2024 */09/2024	Présentation of final demonstrators

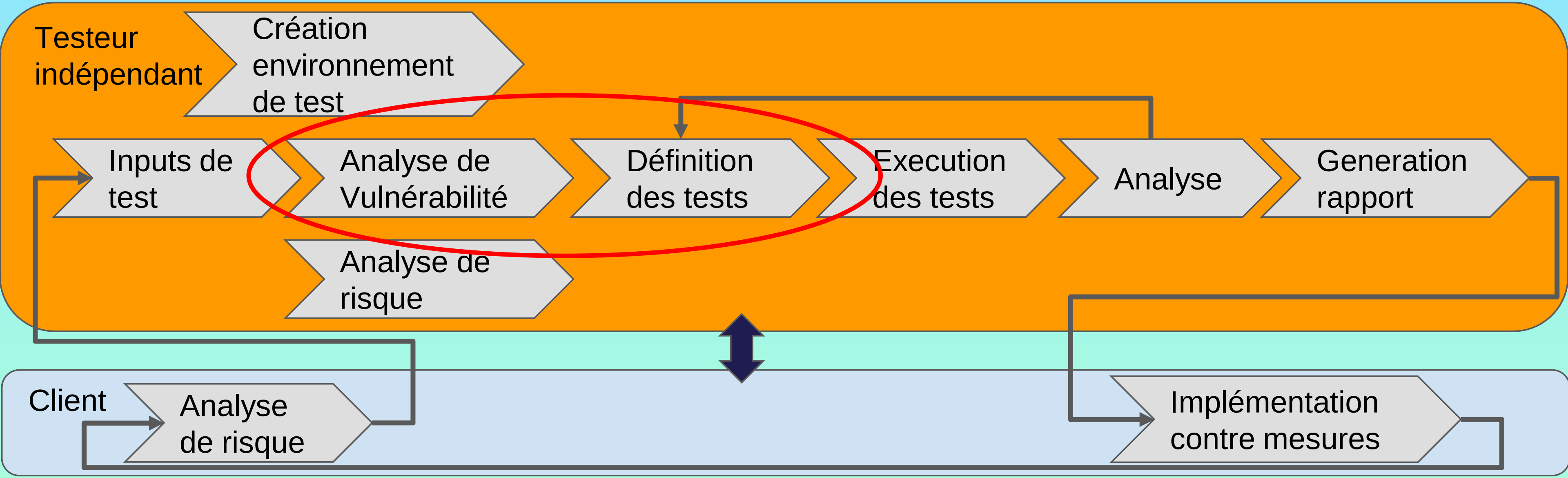
Who participates:

- Companies interested in the challenge
- Challenge Manager
- Researchers contributing to the challenge

Thank you for your attention

Processus de Tests de Pénétration

(https://www.cetic.be/CYRUS-EN)



Testeur indépendant

Création environnement de test

Inputs de test

Analyse de Vulnérabilité

Définition des tests

Execution des tests

Analyse

Generation rapport

Analyse de risque

Client

Analyse de risque

Implémentation contre mesures

Classes Outils

Network mapping with scanner

Database Scan

SAST

DAST & WAST

Fuzzing Tools

Wireless attacks

Compliance assessment

Sniffing

MITM

Vulnerability analysis

Firmware analysis

Password attacks

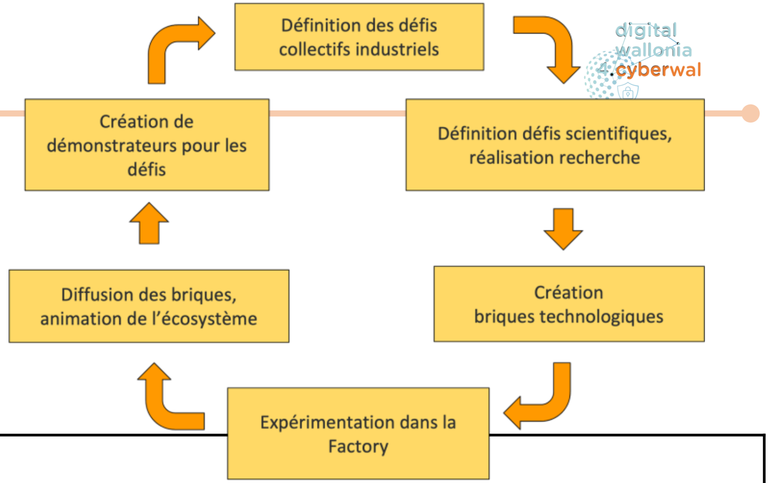
Exploitation tools

Hardware

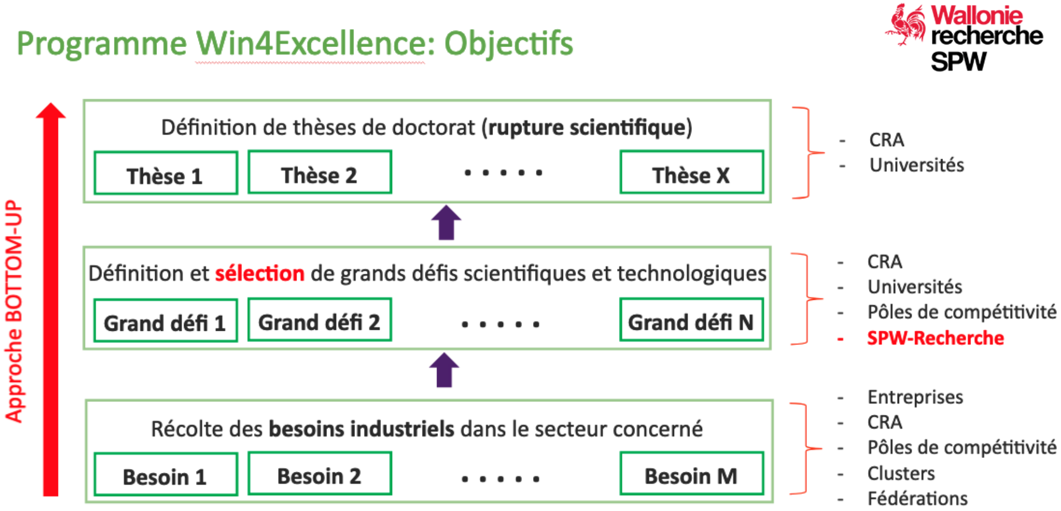
Forensics

Projet CyberExcellence et Défis Collectifs Industriels

- Projet CyberExcellence
  - Projet de recherche en cybersécurité, 01/01/2022, 18,9 millions de budget)
  - Partenaires : 5 universités + 2 CRA
  - Recherche fondamentale mais **au bénéfice du tissu industriel**: réponds aux besoins des entreprises/administrations
- Défi Collectif Industriel
  - Récolte des besoins industriels dans le secteur concerné
  - Identification des défis Collectif Industrie
- Factory
  - Production de briques technologiques
  - Validées dans des démonstrateurs


Définition des défis collectifs industriels
Définition défis scientifiques, réalisation recherche
Création briques technologiques
Expérimentation dans la Factory
Diffusion des briques, animation de l'écosystème
Création de démonstrateurs pour les défis

WP
WP1 : Rendre les systèmes résilients aux cyberattaques : phase de conception.
WP2 : Détection, Réponse, Réaction : Phase Dynamique
WP3 : RGPD et Open data : sécurité à la conception
WP4 : La protection et le partage des données au cœur des préoccupations
WP5 : Laboratoires d'expérimentation, de validation, et d'entraînement
WP6 : Factory et grands défis



Programme Win4Excellence: Objectifs

Wallonie recherche SPW

Définition de thèses de doctorat (**rupture scientifique**)

Thèse 1 | Thèse 2 | . . . . . | Thèse X

- CRA
- Universités

Définition et **sélection** de grands défis scientifiques et technologiques

Grand défi 1 | Grand défi 2 | . . . . . | Grand défi N

- CRA
- Universités
- Pôles de compétitivité
- **SPW-Recherche**

Récolte des **besoins industriels** dans le secteur concerné

Besoin 1 | Besoin 2 | . . . . . | Besoin M

- Entreprises
- CRA
- Pôles de compétitivité
- Clusters
- Fédérations