UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

itee PhD
information technology
electrical engineering


Francesco Crescenzo Grasso

# Binary Code Analysis and AI for Software Security

Tutor: Domenico Cotroneo

Cycle: XXXIX Year: Second

itee PhD
information technology
electrical engineering

# Candidate's information

- MSc degree in **Computer Engineering**
- Research group: **DESSERT**, Dependable and Secure Software Engineering and Real-Time Systems (*dessert.unina.it*)
- PhD start date: **01/11/2023**
- Scholarship type: **PNRR - DM 118/2023**

**DESSERT Research Group**

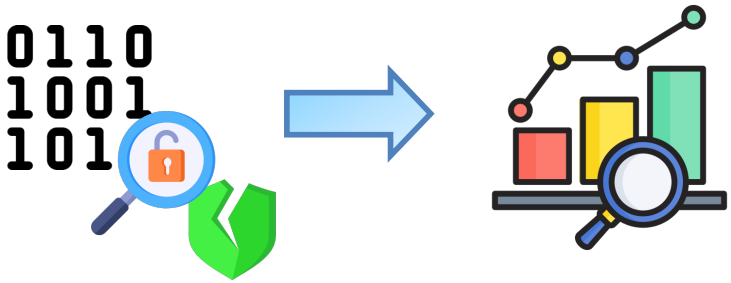
# Summary of study activities

- Ah-hoc PhD Courses:
  - *How to Boost your PhD*

- Conferences/Events Attended:
  - *DSN 2025* (International Conference on Dependable Systems and Networks) Conference at Naples, June 23-26
  - *EuroSys 2025* (European Conference on Computer Systems) Conference at Rotterdam, April 1-3
  - *EuroSec 2025* (European Workshop on Systems Security) security conference located at Rotterdam, March 31, 2025

# Main research summary (1)

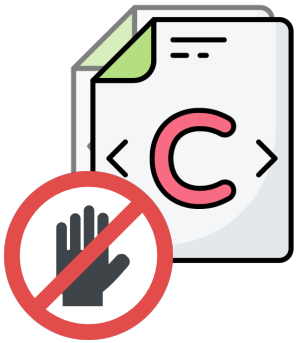
- Binary Code Analysis and AI for Software Security




  - Focus on **machine learning** for **vulnerability prediction**
  - Analysis at **binary** and **source code levels**
  - Intersection of **program analysis, reverse engineering,** and **cybersecurity**
  - GOAL: improve security of *real-world software*

# Main research summary (2)

- Why this research is needed
  - Many systems are released **only as binaries**
  - Source code often **unavailable** (third-party / legacy)
  - Existing datasets are **synthetic** and of **poor quality**



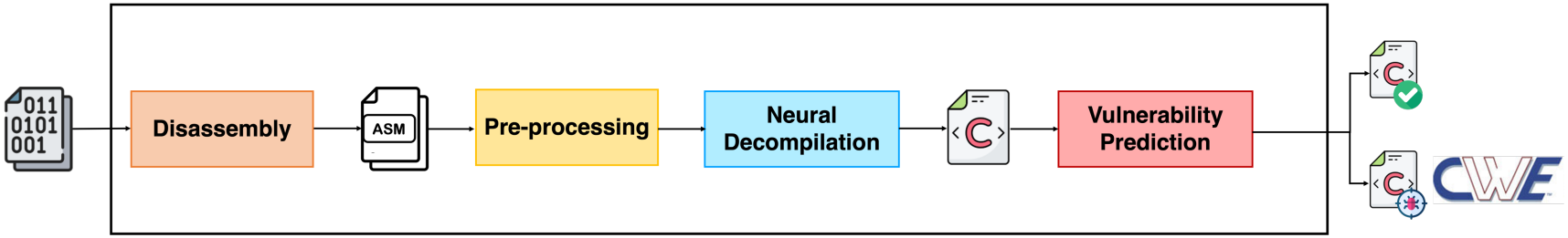


- Research vision
  - **Bridge the gap** between assembly and source code
  - **Detect vulnerabilities** directly from binaries
  - Build **realistic datasets** and ML models for:
    - Binary understanding
    - Decompiled code analysis
    - Vulnerability prediction

# Research Area (1)

- The proposed **methodology** leverages a pipeline whose steps include:
  1. **Disassembly**: extract assembly code from binary files
  2. **Decompilation**: translate assembly back to C/C++ source code
  3. **Vulnerability Detection**: classify decompiled source as vulnerable or safe

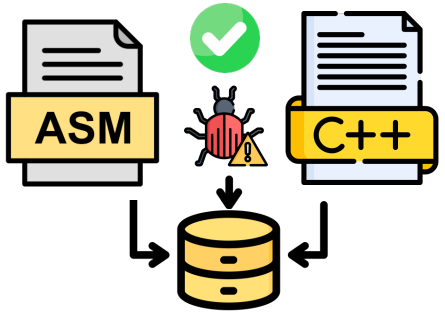
| Binary | ASM x86 | Decompiled C++ | Vulnerability |
|---|---|---|---|
| 0101<br>1001<br>0111<br>... | <funct0>:<br>endbr 64<br>push rbp<br>... | void funct0(){<br>int data;<br>data=0;<br>... | • The function is **vulnerable**. |




| | |
|---|---|
| [P1] | V. Orbinato, F. Grasso, R. Natella, D. Cotroneo,<br>*Can Neural Decompilation Assist Vulnerability Prediction on Binary Code?* ,<br>**ACM European Conference on Computer Systems** (EuroSys Fall 2025) |

# Research Area (2)

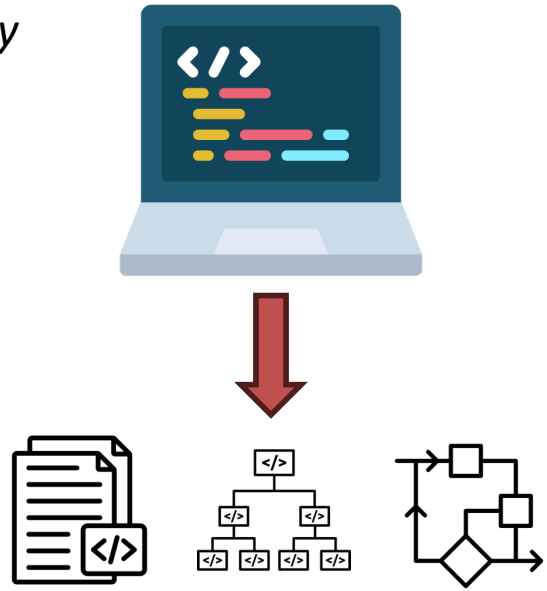
- Building a **real-world dataset**
  1. Collection of **C/C++ source code** and corresponding **ARM/x86 assembly**
  2. Inclusion of both *vulnerable and safe* code samples
  3. Focus on **real-world** projects and realistic compilation settings
  4. Dataset designed to train and evaluate systems that:
     - Reconstruct high-level code from binaries
     - Identify potential vulnerabilities




| | |
|---|---|
| [P2] | V. Orbinato, F. Grasso, R. Natella, D. Cotroneo,<br>*VulDec: A Realistic Dataset for Vulnerability Analysis via Neural Decompilation,*<br>Ongoing work |

# Research Area (3)

- Enhancing Vulnerability Detection Through Multiple Code Representations
  - Detecting issues like *buffer overflows* and *memory corruption* is crucial for software security.
  - Traditional tools: high false positives, poor generalization.
  - ML/NLP-based approach using **multiple representations**:
    - AST, CFG, and raw code (via Joern, Clang)
  - Different views capture **complementary patterns**.
  - Outputs combined through an Ensemble Decision Mechanism
    → **Majority Voting** and **Logical Gates**



| | |
|---|---|
| [P3] | F. Grasso, R. Natella, D. Cotroneo,<br>*Enhancing LLM-Based Vulnerability Detection Through Multiple Code Representations,*<br>Ongoing work |

# Tutorship

- «Laboratorio di Programmazione» BSc course support. Lessons on:
  - Python data collection
  - Python functions
  - Data science
  - Machine learning
  - Neural networks