PhD in Information Technology and Electrical Engineering
Università degli Studi di Napoli Federico II

PhD Student: Cristina Improta

Cycle: XXXVIII

Training and Research Activities Report

Year: First

Cristina Improta

Tutor: prof. Domenico Cotroneo

Domenico Cotroneo

Date: October 18, 2023

1. Information:

- PhD student: Cristina Improta
- DR number: DR996616
- Date of birth: 01/01/1995
- Master Science degree: Computer Engineering
  University: Università degli Studi di Napoli Federico II
- Doctoral Cycle: XXXVIII
- Scholarship type: UNINA
- Tutor: Domenico Cotroneo

2. Study and training activities:

Activity	Type[1]	Hours	Credits	Dates	Organizer	Certificate[2]
Connecting the dots: Investigating an APT campaign using Splunk	Seminar	2	0.4	11/11/22	Proff. S. P. Romano, R. Natella	Y
Cybercrime and Information Warfare: National and International Actors	Seminar	2	0.4	18/11/22	Proff. S. P. Romano, R. Natella	Y
Privacy and Data Protection	Seminar	2	0.4	22/11/22	Proff. S. P. Romano, R. Natella	Y
Digital Forensics	Seminar	2	0.4	06/12/22	Proff. S. P. Romano, R. Natella	Y
IoT Data Analysis	Course	12	4	09/01/23-27/01/23	Dr. Raffaele Della Corte (DIETI)	Y
Using Deep Learning Properly	Course	10	4	10/01/23-24/01/23	Dr. Andrea Apicella (DIETI)	Y
Multi-robot Control of Heterogeneous Herds	Seminar	1	0.2	16/02/23	Dr. Eduardo Montijano	Y

Virtualization technologies and their applications	**Course**	**20**	**5**	**30/01/23-03/03/23**	**Dr. Luigi De Simone, (DIETI)**	**Y**
Analysis and control of functional brain networks	**Seminar**	**1**	**0.2**	**09/03/23**	**Dr. Simone Mancini, Dr. Giacomo Ascione, Dr. Francesco Bajardi**	**Y**
How to Publish Under the CARE-CRUI Open Access Agreement with IEEE	**Seminar**	**1.5**	**0.3**	**05/04/23**	**CARE-CRUI and IEEE**	**Y**
2023 Spring School on Transferable Skills	**Doctoral School**	**9.5**	**2**	**24/05/23-25/05/23**	**Proff. Pasquale Maffia, Valeria Costantino**	**Y**
Open-source software e sicurezza della software supply chain	**Seminar**	**2**	**0.4**	**08/06/23**	**Prof. Roberto Natella**	**Y**
Exploring Advanced Aerial Robotics: A Journey into Cutting-Edge Projects and Neural Control	**Seminar**	**1**	**0.2**	**29/06/23**	**Julien Mellet**	**Y**
Models of human motor coordination – a critical assessment and some open problems	**Seminar**	**1**	**0.2**	**29/06/23**	**Scuola Superiore Meridionale**	**Y**
BGP & Hot-Potato Routing: graceful and optimal convergence in case of IGP events	**Seminar**	**1**	**0.2**	**30/06/23**	**Prof. Valerio Persico**	**Y**
3rd International Software Engineering Summer School	**Doctoral School**	**15**	**3**	**11/09/23-13/09/23**	**Proff. Gabriele Bavota, Michele Lanza**	**Y**

2.1. Study and training activities - credits earned

	Courses	Seminars	Research	Tutorship	Total
Bimonth 1		1.6	8	0.24	9.84
Bimonth 2	8	0.2	4		12.2
Bimonth 3	5	0.5	5		10.5
Bimonth 4	2	1	7		10
Bimonth 5			10		10
Bimonth 6	3		6		9
Total	18	3.3	40	0.24	61.54
Expected	30 - 70	10 - 30	80 - 140	0 – 4.8	

3. Research activity:

The main topic of my research activity concerns the *trustworthiness* of AI code generators, i.e., AI-based solutions to automatically generate source code starting from natural language (NL) code descriptions. The widespread adoption of these AI coding assistants in numerous software engineering tasks, including automatic exploit generation (AEG) for offensive security purposes, calls for the definition of a novel methodology to assess and enhance their usability in real-world scenarios. Indeed, despite the efforts of the community to constantly improve these models, AI code generators still have limitations and potential drawbacks. For example, they may not always generate *correct* code, i.e., code that meets the requirements specified in the NL description, as they may struggle with more complex programming tasks or ambiguous code descriptions. Furthermore, AI models are exposed to a wide variety of security issues targeting both their learning and inference process. For this reason, the use of code generators by AI practitioners and developers, unaware of their pitfalls, potentially leads to the release of buggy, vulnerable software. Therefore, assessing properties such as the robustness, performance and security of AI code generators becomes a crucial challenge.

At the start of the year, my research focused on assessing and enhancing the **robustness** of AI code generators to *perturbations* in the NL code descriptions, i.e., slight deviations in the descriptions caused by the different writing styles of developers. Indeed, developers may have different levels of expertise or use different terminology to describe the same code snippet. Moreover, while some developers use precise specifications, others may provide high-level or abstract descriptions to speed up the coding process. To be used in real scenarios, AI code generators must be robust to the variability of natural language and generate correct code when presented with equivalent descriptions.

We proposed a method that generates new perturbed NL descriptions that diverge from the original ones due to the use of new terms, by performing word substitution, or because they miss part of them, by performing word omission. Then, we applied our method to assess whether the perturbed code descriptions affect the performance of AI code generators in terms of code correctness. Finally, using our method we performed training data augmentation to increase the model's robustness to perturbed code descriptions.

Since an underrated aspect of code generation is the assessment of the quality and **correctness** of AI-generated code, in the later stages of my research I focused on automating the code evaluation process. The golden standard for this analysis is the manual evaluation. However, human assessment is often unfeasible due to the massive amount of data to analyze, which makes it time-consuming and prone to human errors. To address this issue, the literature proposed numerous *textual similarity metrics*, i.e., automatically computed metrics that assess the textual similarity between the generated code and a reference snippet. However, there is no clear indication of which metric is best suited for the code generation task. Therefore, we performed an experimental analysis to understand which metric, among the popularly used ones, is the most correlated to the human evaluation, i.e., the golden standard. Furthermore, we proposed a new method to automatically evaluate the correctness of low-level, *security-oriented* code, i.e., assembly code, that does not require any human intervention. The technique leverages symbolic execution to estimate whether the behavior of the AI-generated program is semantically equivalent to that of a reference program.

Finally, in the last part of the year, my research focused on exploring the **security** implications of using AI models as coding assistants. These AI solutions are strongly data-driven, hence, they rely on large amounts of training data to learn patterns between natural language and programming code. Since collecting the training data is expensive and time-consuming, developers frequently download datasets from untrusted online sources. This exposes AI code generators to a wide variety of security threats, which attracts attackers to exploit their vulnerabilities. Indeed, a concerning issue is *data poisoning,* i.e., the corruption of training data by injecting small amounts of maliciously crafted samples.
To address this threat, we devised a novel data poisoning strategy that injects vulnerable code into the training data. We poisoned AI models and evaluated their susceptibility to data poisoning by assessing the generated code snippets, both in terms of correctness and the presence of security defects.
Finally, we developed a tool to automatically perform vulnerability detection on the AI-generated code.

4. Research products:

P. Liguori, **C. Improta**, S. De Vivo, R. Natella, B. Cukic, D. Cotroneo. “Can NMT Understand Me? Towards Perturbation-based Evaluation of NMT Models for Code Generation”. *IEEE/ACM 1st International Workshop on Natural Language-Based Software Engineering (NLBSE)*, 2022.
Status: published.

P. Liguori, **C. Improta**, R. Natella, B. Cukic, D. Cotroneo. “Who evaluates the evaluators? On automatic metrics for assessing AI-based offensive code generators”. *Expert Systems with Applications Journal (ESWA)*, 2023. Status: published.

C. Improta. “Poisoning Programs by Un-Repairing Code: Security Concerns of AI-generated Code”. *1st IEEE International Workshop on Reliable and Secure AI for Software Engineering (ReSAISE23), co-located with ISSRE23*. 2023. Status: published.

R. Natella, P. Liguori, **C. Improta**, B. Cukic, D. Cotroneo. “AI Code Generators for Security: Friend or Foe?”, *IEEE Security & Privacy*, 2023. Status: under 2nd stage of review.

C. Improta, P. Liguori, R. Natella, B. Cukic, D. Cotroneo. “Assessing and Enhancing Robustness of AI Offensive Code Generators Via Natural Language Perturbations”, *ACM International Conference on the Foundations of Software Engineering (FSE 2024),* 2023. Status: submitted.

D. Cotroneo, **C. Improta**, P. Liguori, R. Natella. “Automating the Correctness Assessment of AI-generated Code for Security Contexts”. *Journal of Systems and Software (JSS)*, 2023.
Status: submitted.

5. Conferences and seminars attended

34th IEEE International Symposium on Software Reliability Engineering Conference (ISSRE 2023). October 9-12 2023, Florence, Italy.
I presented the workshop paper “Poisoning Programs by Un-Repairing Code: Security Concerns of AI-generated Code” to the 1st IEEE International Workshop on Reliable and Secure AI for Software Engineering (ReSAISE 2023).

6. Activity abroad:

None.

7. Tutorship

Tutorship for the “Sistemi Operativi” BSc course. Tutor: Prof. Domenico Cotroneo. Total: 6 hours.

- Process scheduling in Linux
- Inter-Process Communication in Linux
- POSIX threads