PhD in Information Technology and Electrical Engineering

Università degli Studi di Napoli Federico II

PhD Student: Pietro Liguori

XXXIV Cycle

Training and Research Activities Report – Second Year

Tutor: Domenico Cotroneo – co-Tutor: Roberto Natella

1. Information

PhD candidate: Pietro Liguori
Date of birth: 07/09/1989
Master Science title: Master degree in Computer Engineering (cum laude), University of Naples Federico II
Doctoral Cycle: XXXIV
Fellowship type: PhD student grant (Grant Type: Academic)
Tutors: Prof. Domenico Cotroneo, Prof. Roberto Natella
Year: Second

I received my MS degree (cum laude) in Computer Engineering from the Univesità degli Studi di Napoli Federico II in July 2018.

My master thesis focused on anomaly detection in distributed and complex systems such as cloud computing infrastructures. To limit the massive number of false positives (due to the non-determinism of distributed systems), I developed a novel approach of anomaly detection based on a probabilistic model.

I'm currently in the second year of the Ph.D. program in Information Technology and Electrical Engineering (ITEE) at Federico II University of Naples, under the supervision of Prof. Domenico Cotroneo and Prof. Roberto Natella.

2. Study and Training activities

In this second year I attended the following courses and seminars.

Title	Type	Hours	Credits	Dates	Organizer	Certificate
Software Security per Sistemi Industriali	Master Science Course (ING-INF/05)		3	2nd semester 2019-2020	Università degli Studi di Napoli Federico II	Yes
Virtualization technologies and their applications	Ad-hoc		3	April, May 2020	Università degli Studi di Napoli Federico II	Yes
5G International PhD School 2019	Doctoral School		4.3	December 3-5, 2019	Consorzio Nazionale Interuniversitario per le Telecomunicazioni	Yes
Research Strands in Computational Human Behavior Modeling	Seminary	2	0.4	January 24, 2020	University of North Carolina at Charlotte	Yes
Specification-driven Moving Target Defense Synthesis	Seminary	2	0.4	January 22, 2020	University of North Carolina at Charlotte	Yes
An NLP Perspective on Offensive Content in Social Media	Seminary	2	0.4	February 24, 2020	University of North Carolina at Charlotte	Yes
Machine learning for images in space and time: towards real-word applications	Seminary	2	0.4	February 25, 2020	University of North Carolina at Charlotte	Yes
Fingerprinting Human Expression for Computational Health, Wellness, and Creativity	Seminary	2	0.4	February 26, 2020	University of North Carolina at Charlotte	Yes

Hands-on Learning Experiences for Cyber Threat Hunting Education	Seminary	2	0.4	February 27, 2020	University of North Carolina at Charlotte	Yes
Deep Neural Networks for Structured Prediction	Seminary	2	0.4	February 27, 2020	University of North Carolina at Charlotte	Yes
Machine learning models for time series forecasting and question answering on medical data	Seminary	2	0.4	February 28, 2020	University of North Carolina at Charlotte	Yes

	Credits year 1								Credits year 2								Credits year 3									
		1	2	3	4	5	6			1	2	3	4	5	6			1	2	3	4	5	6			
	Estimated	bimonth	bimonth	bimonth	bimonth	bimonth	bimonth	Summary	Estimated	bimonth	bimonth	bimonth	bimonth	bimonth	bimonth	Summary	Estimated	bimonth	bimonth	bimonth	bimonth	bimonth	bimonth	Summary	Total	Check
Modules	25	0	2,2	6	9	3,6	4,8	26	10	4,3	0	0	6	0	0	10	0							0	36	30-70
Seminars	5	0,8	0	0,5	3,8	0,8	0	5,9	4	0	3,2	0	0	0	0	3,2	1							0	9,1	10-30
Research	30	9,2	7,8	3,5	0	5,6	5,2	31	46	5,7	6,8	10	4	10	10	47	59							0	78	80-140
	60	10	10	10	13	10	10	63	60	10	10	10	10	10	10	60	60	0	0	0	0	0	0	0	123	180

3. Research activity

Title: Failure Mode Analysis in Cloud Computing Infrastructures

Description and Study

Cloud computing is becoming an attractive solution for running services with high-reliability requirements, such as in the telecom and healthcare domains.

However, cloud computing systems are often exposed to unpredictable failure conditions. These failures can propagate across several components or layers of the system (e.g., storage, virtual network, compute instances, etc.) in complex ways, leading to cascading effects (**failure propagation**) that make recovery actions more problematic. Therefore, identifying and analyzing failure propagation is an important activity to design more effective recovery actions.

Fault injection is a relevant approach, which emulates faults to anticipate worst-case scenarios, such as network partitions, high network latency, replica crashes, and I/O exceptions. Fault injection has reached a level of maturity that it is routinely used to reveal failures in real-world systems, including cloud computing software such as key-value data stores and distributed computing frameworks (e.g., Cassandra, ZooKeeper), entire cloud computing services (e.g., streaming services deployed by Netflix) and infrastructures (e.g., IaaS providers such as Amazon).

Nevertheless, there are still open issues for its adoption in cloud systems. Indeed, as the scale and the complexity of these systems increase, it becomes harder for developers to identify (and to analyze) failures that are triggered by fault injection. In the current approaches, the system designers write failure specifications before the experiments, and then they look for occurrences of these failures within the experimental data. For example, the most sophisticated approaches check formal specifications over events

and outputs, such as finite state machines, temporal logic predicates, relational logic, and special-purpose languages.

Since these specifications are based on prior knowledge and past experience of system designers about failures, they are not meant for discovering new, unknown failure modes of a distributed system that are missed by the failure specifications. Moreover, writing failure specifications is a time-consuming and cumbersome task, which makes fault injection less applicable.

The current state of practice is to detect failures (e.g., service unavailability, performance degradation) by monitoring the quality of service during the fault injection test; more sophisticated solutions detect failures by monitoring properties expressed with formal specifications, such as finite state machines, relational logic, and special-purpose languages.

However, once a service failure has been triggered by fault injection and detected by monitoring mechanisms, a human analyst still needs to analyze the chain of events (e.g., messages) that occurred among the location where the fault/error is injected and the component that experiences the service failure. Yet, this failure analysis still relies on intuition and manual effort of the human analyst. Unfortunately, manual analysis is too difficult and time-consuming, because of:

- The **high volume of messages** generated by large distributed systems that the human analyst needs to scrutinize;
- The **non-determinism in distributed systems**, in which the timing and the order of messages can unpredictably change even if there is no failure, which introduces noise in the analysis, and increases the effort of the human analyst to pinpoint the failure (i.e., to discriminate the anomalies caused by a fault from genuine variations of the system);
- The use of **off-the-shelf software components**, either proprietary or open-source (such as application frameworks, middleware, data stores, etc.), whose events and protocols can be difficult to understand and to manually analyze.

My research proposal aims to provide automated support for analyzing failures triggered by fault injection in cloud computing systems. I aim to avoid the human analyst to manually inspect thousands of events, by automatically identifying the few relevant events that are related to the injected fault, while discarding noisy, uninteresting events. To this goal, I worked on an approach that extends fault injection, by combining it with black-box tracing and anomaly detection for failure analysis. The driving idea is to train a ***probabilistic model*** of the events in the distributed system under test under *fault-free* conditions, by using variable-order Markov Models for analyzing event sequences. Afterward, the system is tested with fault injection, and event traces are collected under these *faulty* conditions. The faulty event traces are analyzed with anomaly detection by using the probabilistic model, and the anomalous events are reported to the human analyst for understanding how to avoid failures.

I experimentally evaluated the approach in the context of the OpenStack cloud management platform, which is the basis for many commercial cloud management products, and it is widespread both among public cloud infrastructure providers and private users.

The results show that the approach achieves a hit rate higher than 90%. The hit rate saturates around 98% when the probabilistic model is trained with 20 fault-free traces. This size for the training set can be considered small enough for practitioners to apply the proposed approach. Moreover, the proposed approach reduces the false amount rate from ~41% to ~22%. This result means that the probabilistic model can discard many of the differences that are caused by non-deterministic behavior, even if a moderate number of false alarms still needs to be tolerated by practitioners.

In order to help the human analyst to ease the discovery and interpretation of failure modes, I am applying unsupervised machine learning on execution traces of the injected system. In particular, I apply ***clustering*** techniques, built on top of the anomaly detection algorithm, for automatically identifying the classes of failure modes among the tests in a fault injection campaign.

To further help the human analyst at analyzing failures, I worked on a novel technique that clusters fault injection experiments according to classes of failure modes. I evaluated both the ability to identify the number of classes in the data (i.e., how many distinct failure modes occurred in the experiments), and to assign the fault injection experiments to the classes (i.e., the failure mode to which an experiment belongs to). The results showed that the technique can achieve high accuracy under different conditions.

4. Products

4.1 Publications

In this second year, I have produced the following products.

Conference Paper

1. D. Cotroneo, L. De Simone, P. Liguori and R. Natella, "ProFIPy: Programmable Software Fault Injection as-a-Service," *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Valencia, Spain, 2020, pp. 364-372, doi: 10.1109/DSN48063.2020.00052.
2. Cotroneo, Domenico, Luigi De Simone, Pietro Liguori, Roberto Natella, and Angela Scibelli. "Towards Runtime Verification via Event Stream Processing in Cloud Computing Infrastructures", *International Workshop on Artificial Intelligence for IT Operations*, 14 December 2020

Journal Paper

1. D. Cotroneo, L. De Simone, P. Liguori and R. Natella, "Fault Injection Analytics: A Novel Approach to Discover Failure Modes in Cloud-Computing Systems" *in IEEE Transactions on Dependable and Secure Computing,* vol. , no. 01, pp. 1-1, 5555.
   doi: 10.1109/TDSC.2020.3025289
   url: https://doi.ieeecomputersociety.org/10.1109/TDSC.2020.3025289

5. Conferences and Seminars

5.1 Presentations made

I have been invited as a speaker by Dr. Haryadi Gunawi at the University of Chicago, Chicago, USA, for the talk "*How Bad Can a Bug Get? An Empirical Analysis of Software Failures in the OpenStack Cloud Computing Platform*", July 30, 2020

6. Activity abroad

Since January 2020, I am the **University of North Carolina at Charlotte** (**UNCC**), North Carolina, USA, for my abroad period. At UNCC, I am working on a new and innovative research area under the supervision of Dr. Bojan Cuckic and Dr. Samira Shaikh. This new activity is focused on the automatic generation of

software exploits starting from natural language description by using the **Neural Machine Translation** (**NMT**) techniques.

7. Tutorship

In this second year I have been teaching assistant for the course of “Impianti di Elaborazione”, ING-INF/05, Academic Year 2019/2020

I have been co-advisor of the following thesis:

1. “*Runtime Verification Via Stream Processing in Cloud Computing Infrastructures*”, Angela Scibelli, MSc Thesis, Academic Year 2019/20
2. "*Enhancing Failure Analysis of Cloud Infrastructures by Using Deep Learning*", Gabriella Karamanolis, MSc Thesis, Academic Year 2019/20