Finanziato dall'Unione europea
NextGenerationEU

Ministero dell'Università e della Ricerca

Segretariato Generale

Direzione Generale della Ricerca

PRIN: PROGETTI DI RICERCA DI RILEVANTE INTERESSE NAZIONALE – Bando 2022
Prot. 202233YFCJ

## PART A

### 1. Research project title

Smartitude: Automated Testing and Security Assessment of Smart Contracts

### 2. Duration (months)

24 months

### 3. Main ERC field

PE - Physical Sciences and Engineering

### 4. Possible other ERC field

### 5. ERC subfields

| | |
|---|---|
| 1. | PE6_3 Software engineering, programming languages and systems |
| 2. | PE6_7 Artificial intelligence, intelligent systems, natural language processing |
| 3. | |

### 6. Keywords

| nº | Testo inglese |
|---|---|
| 1. | Blockchain |

| | | |
|---|---|---|
| | 2. | Smart Contracts |
| | 3. | Software Testing |
| | 4. | Software Security |
| | 5. | Fault Modelling |
| | 6. | Model Checking |

## *7. Principal Investigator*

| | |
|---|---|
| **DI NUCCI**<br>(Surname) | **DARIO**<br>(Name) |
| **Ricercatore a t.d. - t.pieno (art. 24 c.3-b L. 240/10)**<br>(Qualification) | |
| **03/09/1988**<br>(Date of birth) | **DNCDRA88P03E335W**<br>(Personal identification code) |
| **Università degli Studi di SALERNO**<br>(Organization) | |
| **+393333403254**<br>(Phone number) | **ddinucci@unisa.it**<br>(E-mail address) |

## *PI - Certified E_mail (CEM)*

ddinucci@pec.it

## *Age limits derogation*

Both the PI and the substitute are under 40;

## *8. List of the Research Units*

| nº | Associated Investigator | Qualification | University/ Research Institution | Registered office (address) | Operating office (address) | e-mail address |
|---|---|---|---|---|---|---|
| 1. | DI NUCCI Dario | Ricercatore a t.d. - t.pieno (art. 24 c.3-b L. 240/10) | Università degli Studi di SALERNO | Via Giovanni Paolo II, 132 - FISCIANO - SALERNO (SA) | City: Fisciano (SA) Address Via Giovanni Paolo II, 132 | ddinucci@unisa.it |

| | | | | | | |
|---|---|---|---|---|---|---|
| 2. | BUDDE Carlos Esteban | Ricercatore a t.d. - t.pieno (art. 24 c.3-a L. 240/10) | Università degli Studi di TRENTO | Via Calepina, 14 - TRENTO (TN) | City: Trento (TN) Address Via Calepina, 14 | carlosesteban.budde@unitn.it |
| 3. | CECCATO Mariano | Ricercatore a t.d. - t.pieno (art. 24 c.3-b L. 240/10) | Università degli Studi di VERONA | Via dell'Artigliere, 8 Pal. Giuliari - VERONA (VR) | City: Verona (VR) Address Strada le Grazie 15 | ceccato.mariano@gmail.com |

## *9 - Substitute Principal Investigator (To be identified among one of the associated investigators participating in the project).*

| | |
|---|---|
| **BUDDE**<br>(Surname) | **CARLOS ESTEBAN**<br>(Name) |
| **Ricercatore a t.d. - t.pieno (art. 24 c.3-a L. 240/10)**<br>(Qualification) | |
| **15/01/1988**<br>(Date of birth) | **BDDCLS88A15Z600N**<br>(Personal identification code) |
| **Università degli Studi di TRENTO**<br>(Organization) | |
| **+390461282098**<br>(Phone number) | **carlosesteban.budde@unitn.it**<br>(E-mail address) |

## *Substitute PI - Certified E_mail (CEM)*

cbudde@pec.it

## *10. Brief description of the proposal*

Decentralized finance (DeFi) is changing finance, having prominence from the summer of 2020 with yield farming and the rise of tokens promising attractive yields to crypto traders. According to DeFi Pulse, the total value of cryptocurrencies locked in DeFi protocols is over $78 billion in February 2022 — a growth of 10x since May 2020.

DeFi is based on blockchain, a decentralized, distributed digital ledger, where transactions are used to transfer cryptocurrency from one address to another and deploy and interact with blockchain smart contracts (BSC). The programming model brought in by these software components is relatively novel and peculiar. Once passed the consensus check and written to the blockchain, a BSC and its transactions are immutable, despite programming defects, security vulnerabilities, or faulty transactions. Incorrect computations are frozen forever in the blockchain. Therefore, achieving accurate and comprehensive testing and verification of BSC failures and security vulnerabilities requires specific approaches that consider the persistent nature of code in this novel programming model.

Smartitude aims to provide a comprehensive set of software engineering methods and tools to improve the BSC quality assurance by leveraging automated test case generation and formal methods. More specifically, Smartitude aims to: (i) provide a model of faults occurring during BSC software development, comprising testing, security, and maintainability issues; (ii) elaborate, develop and assess novel automated approaches to generate test scenarios tailored to the BSC execution model to reveal inconsistencies, defects and runtime errors; (iii) derive symbolic models for the execution of networks of BSC to assess the risk of catastrophic executions that can jeopardize the assets of the BSC owner; (iv) integrate the developed solutions into a common infrastructure to support the development of secure and high-quality BSC.

The Smartitude activities will be carried out through a work plan organized in six work packages (WPs). Three refer to the main technical research threads, while the other three refer to the project management, the integration of the solutions developed in the technical WPs, and the dissemination and exploitation of the results. The research and development activities will feature concrete use cases, including industrial environments leveraging Industry 4.0 and a large set of open-source systems.

The project leverages the solid research records of its participants. While the partners have already collaborated on ongoing projects, their combined expertise will help them reach the foreseen goals.

## *11. Total cost of the research project identified by items*

| Associated Investigator | item A.1 | item A.2.1 | item B | item C | item D | item E | sub-total | Total |
|---|---|---|---|---|---|---|---|---|
| DI NUCCI Dario | 19.500 | 37.500 | 34.200 | 2.000 | 0 | 10.000 | 103.200 | **103.200** |
| BUDDE Carlos Esteban | 20.322 | 35.827 | 33.689 | 0 | 4.000 | 8.000 | 101.838 | **101.838** |
| CECCATO Mariano | 0 | 45.000 | 27.000 | 2.000 | 0 | 10.000 | 84.000 | **84.000** |
| **Total** | **39.822** | **118.327** | **94.889** | **4.000** | **4.000** | **28.000** | **289.038** | **289.038** |

N.B. The Item B and TOTAL columns will be filled in automatically

- item A.1: enhancement of months/person of permanent employees
- item A.2.1: cost of contracts of non-employees, specifically to recruit
- item B: overhead (flat rate equal to 60% of the total personnel cost, A.1+A.2.1, for each research unit)
- item C: cost of equipment, tools and software products
- item D: cost of consulting and similar services
- item E: other operating costs

## *PART B*

### *B.1*

## *1. State of the art*

We report the state of the art for the key aspects treated by Smartitude in the context of blockchain smart contracts (BSC): automated generation of test cases and vulnerability assessment.

**Automated generation of test cases to test BSC**
When developing BSC, even simple implementations lead to many non-trivial problems [Del2015], which do not prevent compilation but leave contracts vulnerable to exploitation or unintended behaviors. Among previous work [And2016,Luu2016,Atz2017] analyzing design flaws in published contracts, Zou et al. [Zou2019] investigated the challenges related to smart contract testing and confirmed that almost half of developers desire tools to verify code correctness. These works motivate the need for new tools to develop and test safe BSC effectively.

Nevertheless, all tools have some limitations: (i) SOLIDITY-COVERAGE [Rea2016], SOLIDITY CHECK [Zha2019], and OYENTE [Luu2016] do not produce test suites; (ii) SMARTSHIELD [Zha2020a] focuses on only three vulnerabilities; (iii) ECHIDNA [Alw2022] and MODCON [Liu2020] require the user to define invariants or models of their code, and (iv) ADF-GA [Zha2020b] is tested on a small BSC subset. CONTRACTFUZZER [Jia2018] automatically creates test suites and works on various BSC. However, it focuses on security issues (e.g., re-entrancy vulnerabilities) rather than the functional invariants that contracts should ensure. It can detect only violations of security properties associated with known vulnerabilities without dealing with the peculiar testing aspects of BSC. Additionally, fuzzing tools are less effective at covering complex functionality than guided search algorithms such as genetic algorithms [Fra2014].

To the best of our knowledge, the only general, extensible framework available for functional BSC testing is SOCRATES [Vig2020], elaborated by one of the proposers of Smartitude. It is based on a federated society of interacting bots, where each bot impersonates a distinct user (or role) in the contract with specific behavior. A simulator iteratively assigns an execution slot to each bot. Oracle specification is also supported as contract invariants (predefined generic invariants are already available). SOCRATES represents the starting point for this research project, as it mainly includes random strategies to generate input values and BSC interactions and quite basic oracles. More advanced test case generation strategies require more accurate oracles to adequately test BSC and reveal the complete set of programming defects that affect BSC.

**Risk and security vulnerabilities for BSC networks via formal methods**

The sensitivity of blockchain to security exploits, e.g., in cryptocurrency applications, has vast theoretical and practical evidence [Wan2021,Gre2019,Mue2021]. Consensus protocols can secure transactions to defend users against some attacks, e.g., man-in-the-middle attacks. *However, they cannot prevent security vulnerabilities in BSC* such as integer overflow and transaction ordering [Liu2022]. These security exploits threaten blockchain operativity, from personal accounts to entire virtual economies like NFTs [Mue2021]. Therefore, they have attracted the attention of industry and academia alike.

There are many approaches to detect security issues in BSC; see [Wan2021] for a recent overview. Due to their effectiveness and guarantees, two of the most prominent approaches are testing—detailed in the previous section, including symbolic execution as in Oyente [Luu2016], Osiris [Tor2018], and EClone [Liu2019]—and recently formal methods—such as Zeus [Kal2018], KEVM [Hil2018], and EtherTrust [Gri2018]. Nevertheless, all these approaches are purely empirical—focusing on finding vulnerabilities in individual BSC—and oblivious to risk propagation, such as how exploits escalate in the BSC network that underlies the blockchain.

Moreover, *BSC are human-written code, so their security vulnerabilities will not cease to appear*. Deployed BSC cannot be modified; hence, their vulnerabilities form part of the current and future scenario, to the point that attack indicators have been devised for decision making [Alm2018]. Nevertheless, a deployed BSC can be deleted, making for a rich case of risk analysis, where an automatic assessment of the vulnerability of deployed contracts—e.g., embedded in the BSC using oracles—promises to improve the overall security of the pool of assets of its owner.

Such risk analysis calls for novel vulnerability assessment models, which Smartitude sets forth to investigate and develop. These models must be sensitive to vulnerability discoveries in BSC since these are the most likely to impact blockchain assets. The nature and number of vulnerabilities will also play a pivotal role, and data-based risk estimation—that stands on the historical nature of attacks and their propagation—will reflect the known behavior of the corresponding blockchain.

---

## *2. Detailed description of the project: methodologies, objectives and results that the project aims to achieve and its interest for the advancement of knowledge, as well as methods of dissemination of the results achieved*

**BLOCKCHAIN SMART CONTRACTS**

A blockchain [Nak2008] can be viewed as a decentralized, distributed digital ledger: an ordered list of blocks containing an ordered list of transactions. New blocks are added by miners, who follow a consensus protocol that dictates the rules of the blockchain, including how to add new data and deal with conflicting versions of the blockchain. On the Ethereum blockchain [Woo2014], transactions can be used to transfer Ether (ETH) cryptocurrency from one address to another and deploy and interact with blockchain smart contracts (BSC). Ether is also used to compensate the miner, who receives a small fee (called Gas) from the transaction sender for registering a transaction on the blockchain. In addition to sending Ether, transactions can also be used to interact with and deploy BSC. Because of its inner working, the Ethereum blockchain can store (almost) any data type, so long as modifications are made in a transaction-based manner. Its creators have leveraged this property to store (compiled) pieces of code, i.e., smart contracts, on the blockchain, which can be used as follows. Each transaction has a "Data" field where a transaction sender can store bytecode. When sending to a previously unused address, this bytecode can be interpreted by miners that use the Ethereum Virtual Machine (EVM) to create a new BSC whose bytecode is stored on the blockchain at the new address. After a contract has been deployed, the transactions sent to its address can invoke the execution of the code stored on the blockchain by including the method to be invoked and any input parameters in the Data field of the transaction. The EVM specifies how to alter the state of the system based on the Data field of the transaction and the code stored at the specified address. A special function called FALLBACK is invoked if a transaction is issued without a recognizable method in its "Data" field. Since writing bytecode by hand is impractical, several high-level programming languages have been created, the most popular of which is Solidity, which is inspired by Python, C++, and Javascript. Solidity BSC are similar to classes in Object-oriented programming and behave similarly to objects: the smart contract code serves as a blueprint to deploy many instances on the blockchain, each with its address and internal state. Similarly, Solidity BSC have both public functions and variables that can be accessed from outside the smart contract and private functions and variables that can only be interacted with by the contract itself.
Besides the technical details of BSC operations in a blockchain, *the programming model brought in by smart contracts is quite novel and peculiar*. Once passed the consensus check and written to the blockchain, a BSC and its transactions are immutable, despite programming defects, security vulnerabilities, or even faulty transactions. *This means that incorrect computations are frozen forever in the blockchain*. Moreover, BCS are subject to a peculiar life cycle. While objects/structures in object-oriented/procedural code are allocated and deallocated, BSC are permanent after their instantiation and typically never deallocated from the ledger. This aspect makes BSC subject to a unique fault model, as discussed in objective O1 below. For these reasons, *achieving accurate and comprehensive testing and verification of BSC failures and security vulnerabilities requires specific approaches that consider the*

*persistent nature of code in this novel programming model*.

**CHALLENGES**

The project aims at finding novel solutions to the following challenges:

- **C1. Lack of a holistic fault model for BSC that includes software defects, vulnerabilities, and quality issues.** Zou et al. [18] noticed that developers lack testing practices specifically tailored to BSC. However, not all bugs are the same. Thus, understanding the nature of faults present in BSC is essential to developing tools to detect functional, security, and quality undesirable effects.

- **C2. Lack of methods for automatically generating test cases to prevent defects in BSC.** Despite the effort devoted in recent years by the research community, Zou et al. [18] showcased several issues related to BSC software dependability. In particular, according to their findings, developers lack tools that consider all corner cases and scenarios. Many tools for BSC testing are only prototypes, and they do not consider BSC test quality and readability.

- **C3. Lack of assessment methods to guarantee BSC security.** Detecting vulnerabilities in BSC has been a hot research topic, especially since the infamous DAO (Distributed Autonomous Organization) attack in 2016, where roughly 60 million dollars worth of Ether were stolen due to an unforeseen exploit in a published smart contract. The proper implementation of BSC is particularly challenging due to specific blockchain properties, such as the immutability of committed blocks and its distributed and decentralized nature. However, as noticed by Zou et al. [18], developers currently have no effective way to assure code security, and tools like code auditing and formal verification techniques are highly desired.

**PROJECT OBJECTIVES**

Smartitude aims to provide a comprehensive set of software engineering methods and tools to improve the BSC quality assurance by leveraging automated test case generation and formal methods.
Zou et al. [Zou2019] report that developers face several challenges during smart contract development, including no effective way to guarantee the security of smart contract code and limited availability of tools for their testing and maintenance. In particular, the project aims at bridging this gap in a four-fold manner:

- **O1. A Fault Model Catalog for BSC.** Smartitude will provide a model of faults occurring during BSC software development. This catalog will target software dependability from different perspectives, particularly testing, security, and maintainability. Based on this taxonomy, we will propose a tool to automatically inject the different faults in bug-free contracts to provide a strong testbed that researchers can rely on and extend in the future to move towards BSC high-quality and security guarantee.

- **O2. Automated Test Case Generation for BSC.** Existing approaches to automatically test procedural code are expected not to be fully effective on BSC due to their specific execution model and environment. Thus, an objective of Smartitude is to elaborate, develop and assess novel automated approaches to generate test scenarios tailored to the BSC execution model, including input values synthesis, generation of interaction sequences, and provisioning of runtime assertions to reveal inconsistencies, defects, and runtime errors.

- **O3. BSC Automated Vulnerability and Risk Assessment.** Smartitude intends to derive symbolic models for the execution of networks of BSC to assess the risk of catastrophic executions that can jeopardize the assets of the BSC owner. These models will be constructed from empirical data—whose availability is guaranteed by the blockchain technology—and are intended to be sensitive to vulnerabilities discovery, with the ultimate aim of automating risk-contention policies as BSC-encoded clauses.

- **O4. Integration of the Developed Solutions.** Smartitude aims at integrating the developed solutions into a common infrastructure to support the development of secure and high-quality BSC.

**METHODOLOGY AND RESULTS**

In the following, we detail the methodology, expected results, assessment methods, and relevance of the results for all objectives previously reported.

**O1. A Fault Model for BSC**
*Methodology*: A cornerstone of the project is a fault model for BSC. The model will target a threefold objective: defects, code smells, and security issues. Although previous work provided approaches to support developers in understanding defect types, this analysis did not focus on the specific BSC characteristics. Understanding bug types would be beneficial to support developers in bug triaging and fixing.
To this end, we will develop a taxonomy representing the bug types that occur during software development and maintenance. First, we will conduct a mining software repository campaign to find relevant bug reports describing software bugs occurring in BSC in the context of public software repositories. We will conduct iterative content analysis sessions [Lid2010] to analyze such reports from the software testing, security, and quality perspectives and assign labels to compose our fault model. We will complement the taxonomy by conducting a multivocal literature review [Gar2019]. Given that the literature on smart contract development is still scarce, we will analyze both academic white literature [Kit2007] and industrial gray literature [Gar2019] to provide a holistic overview of issues affecting the software dependability of these software components.
Finally, following the lines of previous work [Gha2020], we will develop a tool to inject bugs into all potential locations of BSC. Compared to such previous work, we will not only target security issues but also functional and quality-related aspects to provide an integrated tool for researchers in software testing, security, and quality of BSC.

*Results*: The objective will deliver three exploitable assets needed to build a common background on software bugs and vulnerabilities specific to BSC and assess the effectiveness and efficiency of the tools developed in the context of O2 and O3, and their integration performed in O4. In particular, it will deliver (i) a catalog of defects, code smells, and vulnerability issues, (ii) several datasets of software defects and vulnerabilities, (iii) a mutation seeding tool to inject faults tailored explicitly for BSC.

*Assessment*: We will assess the quality of the fault model (i.e., the taxonomy and the dataset of faults) by following de facto standards in academia for content analysis [Lid2010] and literature reviews [Gar2019][Kit2007]. Finally, we will follow the latest guidelines for high-quality software repositories [Mun2017]. Software fault injection methods commonly support researchers and practitioners in assessing software dependability issues. Such methods must produce faults that represent real faults and inject them in an efficient and readable way. Therefore, we will use state-of-the-art methods to assess the validity of the mutation seeding tool. Finally, we will analyze the performance of widely-used static analysis tools and those developed in the context of WP3 and WP4 in detecting the injected bugs, as previously done [Gha2020].

*Relevance of the results*: The project will release a set of exploitable assets essential for any researcher and practitioner interested in understanding software defects and vulnerabilities of BSC and comparing the effectiveness and efficiency of the tools available to verify the presence of such issues.

**O2. Automated Test Case Generation for BSC**
*Methodology*: First of all, we plan to define innovative coverage metrics that capture the specific nature of BSC, befitting the peculiar programming model of blockchain technology. Existing coverage metrics are mainly based on quantifying the extent of code or data/control dependencies exercised by tests. They are inadequate in measuring test completeness regarding the internal contract states that are possibly enabled after peculiar user interactions.

Most of the available testing approaches are purely based on generating random interactions with random input data, so they are expected to exercise only shallow scenarios and reveal a small fraction of actual defects. We plan to elaborate on more advanced approaches to deliver more comprehensive tests, starting from evolutionary-inspired algorithms. This test generation is guided by multi-objective optimization that maximizes test coverage and diversity while keeping tests simple. Subsequently, we will develop novel test case generation strategies exploiting symbolic execution or concrete symbolic execution. They will craft test cases that aim to satisfy specific requirements, such as realizing a currently untested interaction or taking a BSC in a (possible corner case) state that would enable interactions that have not yet been tested.

Eventually, novel oracles will be investigated that can assert the state and the result of the BSC under test. Initially, oracles will be based on those generic invariants (identified by Objective O1) that should always be satisfied by a correct BSC. Subsequently, we will investigate customized (i.e., contract-specific) oracles. These latter oracles will be based on the documented working constraints imposed by those remarkable and recurrent interfaces that BSC often implements, for instance, when realizing tokens or NFT.

*Results*: This objective will deliver a framework to automatically test BSC capable of exploiting (i) effective sequences of interactions; (ii) with proper input values; and (iii) accurate oracle to assess the outcome of the test cases. To reduce the effort spent on engineering details and concentrate on delivering novel research results, an initial existing extensible testing framework, namely SOCRATES, will be used as starting point to build novel contributions on top.

*Assessment*: The novel testing approaches elaborated by the project will be assessed from an experimental and empirical point of view. The benchmark of BCS delivered by Objective O1 will be subject to extensive testing to measure the capability of revealing

known defects, the cost to pay for it (e.g., test case generation time). This same benchmark will be used to compare our approach with the baseline state of the art on common ground to reveal strong and weak points of the compared approaches and would guide future research directions and improvements.
Eventually, our testing framework will be deployed in the Industry 4.0 showcase laboratory at the University of Verona (ICE lab) to test the BSC that is supposed to monitor, enforce and publish industrial production KPIs for quality assessment.

*Relevance of the results*: The testing framework represents a precious result for the growing BSC industry and the testing community.
Once deployed in the blockchain, BSC and their transactions can not be reverted. Programming mistakes caused substantial revenue loss because of malicious interactions or locked digital currency in faulty BSC. Cheap, fast, and automated testing is instrumental in revealing defects before code is deployed in the blockchain and becomes unchangeable.
The testing framework would also represent a precious reusable asset for the research community. Implementing novel research ideas as prototypes is not supposed to waste valuable effort and start implementation from scratch. Instead, researchers could leverage our extensible testing framework and only implement those small incremental contributions that realize their novel algorithms.

**O3. BSC Automated Vulnerability and Risk Assessment**
*Methodology*: The goal of O3 is to generate a tooling framework to model BSC networks from the perspective of formal methods (FM), thus ensuring formally guaranteed outputs. The objective of the models is to allow security-relevant queries, e.g., the probability of containing a specific type of vulnerability or executing clauses related to (other) tainted BSC, its expected execution time.
Such formal models must be written in the language of a formal tool: we will assess existing tools for fitness to our purpose, leveraging the collaboration of Arnd Hartmanns (Univ. of Twente, NL), who has extensive experience in FM tools comparisons—see, e.g., the ARCH-COMP and QComp series [Aba2021, Bud2021]. Investigations include a selected case-study blockchain (probably Ethereum given its BSC environment, also Binance), from which to compute empirical distributions for the probabilities of inserting exploitable BSC security vulnerabilities.
Once the tool is chosen and the distributions learnt, we will design model templates for the most common vulnerabilities found in our catalog, then deploy them to study the security behavior in networks of (models of) BSC via statistical model checking. The framework would ultimately be used to build attack models based on public vulnerability databases to assess the risk—i.e., success likelihood and impact—of different attack profiles.

*Results*: First, the objective will focus on a survey paper concerning the SoA of FM tools to model and assess the security of networks of BSC. The tools include the Modest Toolset, Storm, Uppaal, and PRISM. Once a tool has been chosen, we will formalize the vulnerabilities found in our catalog, developing model templates for the generic deployment of networks of contracts. Second, the objective will provide a software tool and several model templates integrated as a modeling framework for security vulnerability studies. Finally, we will use our framework on real-world vulnerabilities databases to assess the attack risk of existing assets in a set of potential scenarios based on known attack profiles, resulting in an empirical study from existent blockchain environments.

*Assessment*: For the first result, the FM tools will be assessed based on the degree of granularity to which they can model empirical (arbitrary) distributions, their support for both exhaustive and statistical model checking, and the capacity—and ease of use—of their input languages to represent the required vulnerabilities. These vulnerabilities include re-entrancy, integer overflow, and unchecked return values. "Ease of use" will be assessed as the lines of code of the resulting models and the similarity of the modeling language to "standard" programming languages such as C.
The second result includes the actual models of networks of BSC based on the catalog of known vulnerabilities, including the distributions of attack probabilities. Success will be assessed as the number of vulnerabilities whose semantics could be captured in the models and the possibility to model the required distributions for inter-contractual transactions. If such behavior is non-deterministic (e.g., the distributions change entirely for different case studies, even within the same blockchain), the contention plan is to use the model checking—as opposed to simulation—capacities of the tooling framework. This alternative could reduce the size of the representable models but would also change the type of results from statistical (confidence intervals) to fully formal (correct lower and upper bounds).
For the final result, reproducing the risk of a recorded transaction in our modeling framework will serve as validation. We plan to extrapolate to future transactions and measure how our estimations match the real blockchain executions.

*Relevance of the results*: The approach will advance the state of the art in security analysis of BSC, first by providing a technical comparison (from a security-modeling perspective) and suitability analysis of the main tooling frameworks in FM. The formal models will include, for the first time, stochastic distributions that govern inter-contractual transactions. They will push the state of the art by introducing a modular modeling strategy to capture non-deterministic aspects of BSC networks and featuring probabilistic decisions inferred from empirical (past) statistics.

**O4. Integration of the developed solutions**
*Methodology*: A key objective of Smartitude is to integrate the contributions deriving from objectives O1-O3 into an integrated toolset and methodology that allows developers to improve the dependability of BSC and, in particular, its reliability and security. We will conduct iterative integration steps repeated over the project, starting from SOCRATES [Vig2020], the research prototype already elaborated by the project proposers. Periodically, the project will release artifacts executable within Truffle Suite, a development and

testing framework for blockchains using the Ethereum Virtual Machine (EVM).

*Results*: This objective will generate an integrated toolset and methodology that can be used to develop trustworthy BSC.

*Assessment*: The effectiveness of the integration will be periodically assessed and will target the tools integrated at each evaluation stage. We will conduct empirical studies, controlled experiments, and longitudinal studies with a threefold objective (1) understanding the testing and vulnerability assessment capabilities of the integrated tools, (2) estimating the benefits provided by the tools to the quality of the development process, and (3) verifying the degree of confidence of developers when exposed to the use of the released tools.

*Relevance of the results*: The project will release an integrated solution to automatically test and find software vulnerability in BSC, which is not currently available and is expected to ease the overall development of those systems. In addition, the availability of such tools will favor dissemination, ease the impact, and facilitate the adoption of the Smartitude solutions by researchers and practitioners.
The integration of the tools will bridge the gap between both techniques. On the one hand, usually, testing activities achieve partial coverage. Formal verification can be employed where testing is not effective. On the other hand, formal verification frequently requires too complex abstractions. In these cases, testing can be used to check the precision loss due to approximate abstractions, i.e., testing checks the formal verification results.

**CASE STUDIES**
The ICE (Industrial Computer Engineering) Laboratory [https://www.icelab.di.univr.it/] was set up by the Computer Science department of the University of Verona to promote synergy between the technological innovation of Industry 4.0 and the education and research centers located in the Verona area. The facility is part of the IT excellence project, awarded 8 million euros by the Italian Ministry of Education, Universities, and Research. The project involves creating prototypes integrating augmented reality to be used in production lines. The ICE Laboratory is equipped with state of the practice, Industry 4.0 compliant machinery: additive and subtractive manufacturing, assembly and disassembly robotic harms, and quality control are the phases common to many industries.

All this equipment is controlled by a modern software stack to support a high level of reconfigurability of these general-purpose machines to adapt to the changing production requirements typical of Industry 4.0. Automated production adopts a model-driven approach, where instead of manually coding the complex interactions among different machines, the operator specifies the production steps using UML diagrams. UML diagrams are then automatically turned into programs and instructions for the different components of the ICE Laboratory that would cooperate to deliver the programmed production recipe. Besides control instructions, a set of BSC is also automatically generated based on UML models. These Industry 4.0 BSC are responsible for receiving those IoT data generated during automated production, computing quality KPI, publishing, and certifying them to enable subsequent quality assurance by the intended customers or product end-users. Production KPI might range from mechanical quality to green certification (i.e., production energy consumption, waste reduction) of the produced pieces.

Considering the high level of automation in Industry 4.0 BSC, manual quality review and manual testing are not appropriate. Such activities would take too long to be compatible with the fast reconfigurability of an Industry 4.0 production plant. Nevertheless, vulnerabilities and defects are not acceptable since implementation mistakes cannot be patched once BCS are deployed in the blockchain.

In this scenario, fully automated validation and testing of Industry 4.0 BSC is a viable solution pioneered by Smartitude, capable of detecting vulnerabilities and defects and still allowing to meet the key requirements in Industry 4.0 of high and fast reconfiguration.

Finally, we will collaborate with Exrade Srl, a software company based in Trento that develops BSC for several industry sectors. They are interested in supporting the research carried out by Smaritutde, allowing us to conduct surveys and/or access gold standard datasets related to BSC testing. This collaboration will allow Smartitude to acquire further knowledge on BSC testing and vulnerability assessment. In the following, we attached the letter of intent they provided to support the Smartitude proposal.

exrade

**Lettera d'intenti**

*Bando PRIN 2022 - Progetti di Ricerca di Rilevante Interesse Nazionale - Ministero dell'Università e della Ricerca - D.D. n. 74 del 25-01-2022 Manifestazioni di interesse*

**Al Magnifico Rettore**
**Prof. Pier Francesco Nocini**
**Università di Verona**
**Via dell'Artigliere, 8, 37129, Verona**

**Oggetto**: Bando PRIN 2022 - Progetti di Ricerca di Rilevante Interesse Nazionale - Ministero dell'Università e della Ricerca - D.D. n. 74 del 25-01-2022 Manifestazioni di interesse D.M. 10 agosto 2021, n. 1061 - Manifestazioni di interesse. Lettera di intenti per la manifestazione di interesse a collaborare al progetto di ricerca *Smartitude: Automated Testing and Security Assessment of Smart Contracts.*

Il sottoscritto **Luigi Telesca**, in qualità di **LEGALE RAPPRESENTANTE** dell'impresa **Exrade srl**, sussidiaria di Trakti Ltd., con sede legale in **Trento, Via Roberto da Sanseverino n. 95**, CAP 38122 CF e P.IVA 02203850223 , conferma la volontà di collaborare con l'Università degli Studi di Verona all'attività di ricerca dal titolo "*Smartitude* ", relativo al bando in oggetto. In particolare, l'impresa, qualora tale progetto venisse ammesso al finanziamento, valuterà la possibilità di:

- supportare le attività di ricerca previste dal progetto permettendo ai ricercatori coinvolti di condurre survey e/o di accedere ai gold standard dataset utilizzati per il collaudo di sistemi software basati su tecnologia blockchain, allo scopo di acquisire conoscenza sul processo di collaudo di moduli software basati su tecnologia blockchain nonché sulle pratiche di valutazione e miglioramento della qualità di tali sistemi messe in atto dall'azienda;
- accogliere i componenti dell'unità di ricerca dell'Università degli Studi di Verona/Salerno/Trento presso la propria sede operativa in **Trento, Via del Suffragio n. 46** per pianificare/svolgere attività di ricerca previste dal progetto e oggetto del presente accordo.

In ogni caso, l'attività di supporto di cui ai due punti sopra descritti, dovrà essere concordata e disciplinata da successivi accordi tra Exrade srl e l'Università di Verona attraverso la sottoscrizione dei contratti all`uopo necessari e adatti alla fattispecie del caso concreto.

Il referente aziendale è individuato nella persona del dott. Luigi Telesca (luigi@exrade.com) Per l'intera durata del periodo di svolgimento del progetto di ricerca nessun onere sarà a carico dell'impresa, e nessun compenso sarà richiesto all'Università.

Trento, 23/03/2022

Firma e timbro

exrade
**EXRADE s.r.l. a Socio Unico**
Via Roberto da Sanseverino 95
38122 – Trento, Italy
PIVA/VAT: 02203850223

eXrade s.r.l. a Socio Unico
Via R da Sanseverino 95
38122 Trento – Italy

tel. fax. 0461- 527782
email. info@exrade.com
pec. exrade@pec.it

CAP.SOC. € 30.000,00 i.v.
P/IVA VAT: 02203850223
http://www.exrade.com

## 3. Project development, with identification of the role of each research unit, with regards to related modalities of integration and collaboration

The activities of Smartitude are organized into six Work Packages (WPs) to address the objectives defined above. WP1 regards project management. Three WPs (WP2, WP3, WP4) directly map onto the objectives O1, O2, and O3. WP5 concerns the integration and validation of the proposed solution. Finally, WP6 concerns the dissemination of the project results. In the following, we provide a brief description of each WP, indicating its tasks and summarizing the partners' roles and contributions.

**WORK PACKAGE STRUCTURE**

**WP1: Project Management (Leader: UNISA)**
This WP aims at coordinating the project development, ensuring timely progress, and productive coordination and collaboration among the units. The work is organized around two tasks:

- T1.1 - Project development infrastructure setting up (M1-M24; Leader: UNISA; Contributions: all).
- T1.2 - Technical coordination (M1-M24; Leader: UNISA; Contributions: all).

**WP2: Framing a Fault Model for BSC (Leader: UNISA)**
This WP aims to provide a holistic fault model to increase the knowledge concerning the nature of software bugs and vulnerabilities in BSC. Such a catalog will be leveraged to provide a comprehensive dataset of software defects and vulnerabilities and develop a mutation seeding tool to inject software faults in BSC automatically.

- T2.1 - Catalog of defects and vulnerability issues in BSC (M1-M12; Leader: UNISA; Contributions: UNITN).
- T2.2 - Datasets of software defects and vulnerabilities affecting BSC (M7-M18; Leader: UNISA; Contributions: UNIVR).
- T2.3 - Mutation seeding tool to inject faults in BSC (M13-M24; Leader: UNISA; Contributions: UNIVR).

**WP3: BSC Automated Test Case Generation (Leader UNIVR)**
This WP aims to provide test case generation methods and tools to verify BSC. First, such methods will leverage genetic algorithms, symbolic execution, and societies of bots. Afterward, they will consider the fault model to cover a broader selection of software defects and provide automated testing oracles.

- T3.1 - Automated generation of test scenarios (M1-M18; Leader: UNIVR; Contributions: UNISA).
- T3.2 - Automated generation of error test scenarios (M13-M24; Leader: UNIVR; Contributions: UNITN).
- T3.3 - Automated testing oracles based on interface documentation (M7-M24; Leader: UNIVR; Contributions: UNISA).

**WP4: BSC Automated Vulnerability and Risk Assessment (Leader UNITN)**
This WP aims to generate a—formal methods—tooling framework to model BSC networks. The formal models generated will allow security-relevant queries such as inquiring about the exploitation probability of the vulnerabilities present in the fault catalog. These models will also be parameterized to take the input from the testing results from test case generation to adjust the models to specific cases automatically.

- T4.1 - Modeling framework for BSC (M1–M7; Leader: UNITN).
- T4.2 - Formal models of networks of BSC (M7–M15; Leader: UNITN; Contributions: UNIVR).

- T4.3 - Attack models from the blockchain (M14–M24; Leader: UNISA; Contributions: UNITN).

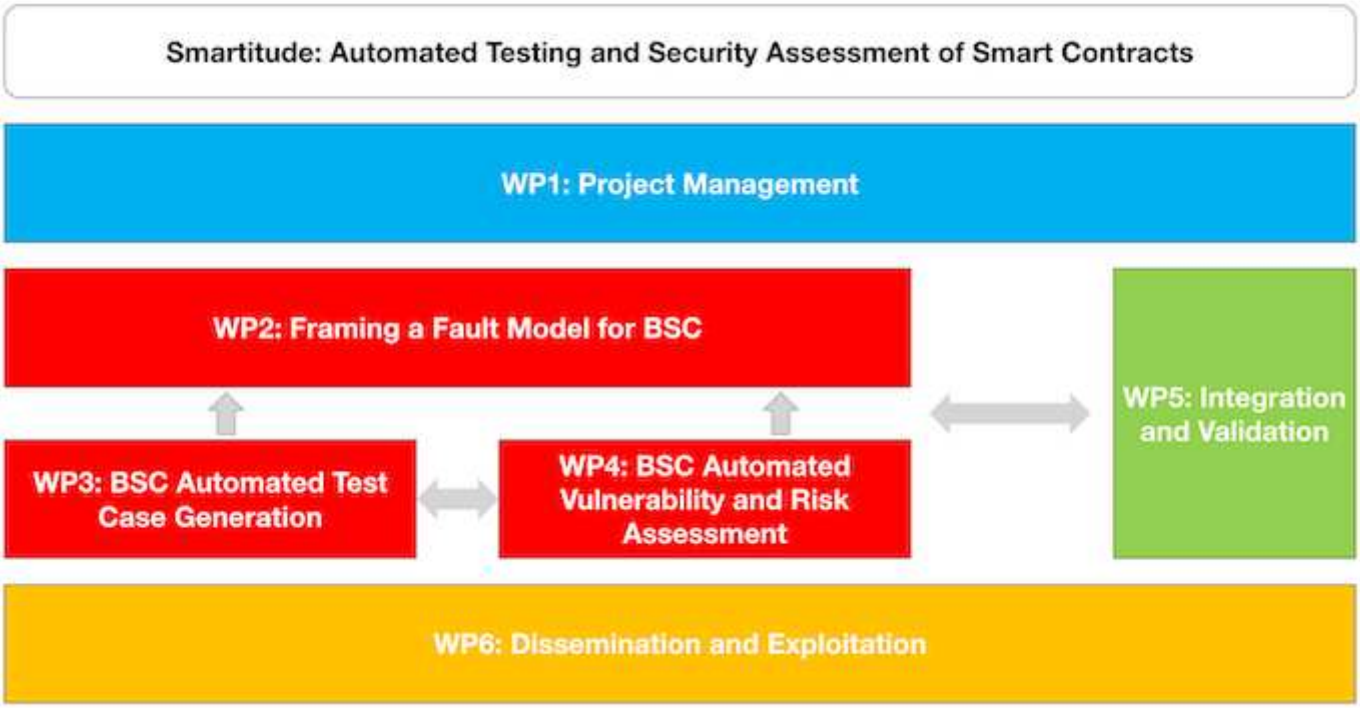
**WP5: Integration and Validation (Leader: UNITN)**
This WP will define an extensible framework capable of integrating the testing and verification techniques elaborated by the project. This research outcome will allow the research community to contribute with their novel research idea. The integrated framework will be exploited in a case study at the ICE laboratory of the University of Verona (department of excellence project) that applies BSC for production certification.
We will combine software testing generation and software vulnerabilities to leverage the characteristics of both techniques to provide more trustable BSC, e.g., by pipelining testing output into formal model parameters.
Finally, we will provide a common public benchmark of faulty and vulnerable smart contracts to assess our techniques and provide a comparison reference to the research community.

- T5.1 - Integration framework (M13-M24; Leader: UNIVR; Contributions: all).
- T5.2 - Combination of testing and formal verification (M19-M24; Leader: UNITN; Contributions: all).
- T5.3 - Experimental assessment (M7-M24; Leader: UNISA; Contributions: all).

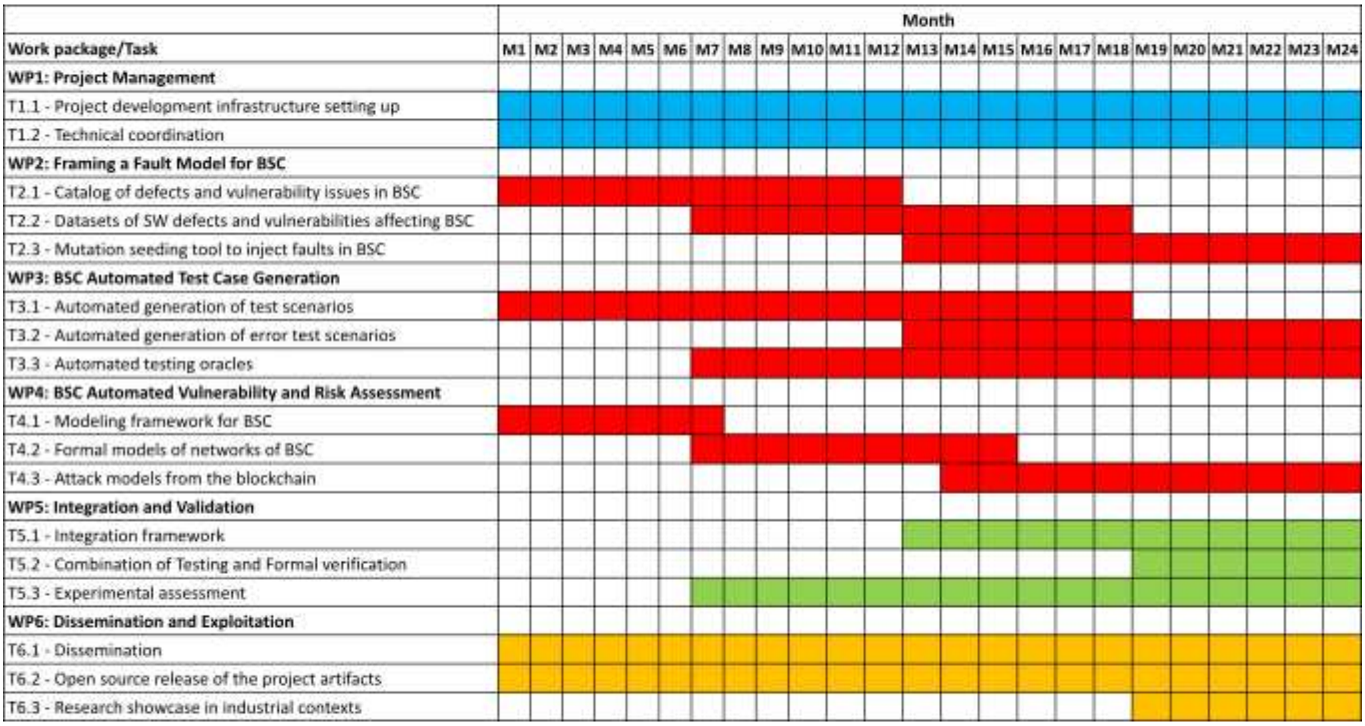
**WP6: Dissemination and Exploitation (Leader: UNIVR)**
This WP will coordinate publication activities and other dissemination activities (website, social media, conferences, workshops, tutorials). It will also drive the project software release and additional material as open source. Finally, we will showcase the exploitable research assets in the context of the ICE lab of the University of Verona, which companies regularly visit.

- T6.1 - Dissemination (M1-M24; Leader: UNISA; Contributions: all)
- T6.2 - Open source release of the project artifacts (M1-M24; Leader: UNITN; Contributions: all).
- T6.3 - Research showcase in industrial contexts (M19-M24; Leader: UNIVR; Contributions: all).

**WP DEPENDENCIES AND DEVELOPMENT METHODOLOGY**




The figure above shows how the technical WPs (WP2, WP3, and WP4) depend on each other. More precisely: the automated test case generation and vulnerability assessment tools will depend on the catalog of defects and vulnerability issues (i.e., T2.1). Furthermore, the datasets of software defects and vulnerabilities (i.e., T2.2) and the mutation seeding tool to inject faults (i.e., T2.3) will be used to conduct empirical experimentation to verify their effectiveness and efficiency (T5.3). The implementation of attack models from the blockchain (T4.3) will take input from the test cases of T3.2. The activities in WP3 and WP4 will start at M1, leveraging a set already available of defects and vulnerabilities that will be integrated later.




The Gantt chart shown in the figure above specifies how the different activities span through the project duration. We depict in blue the management activities (WP1) while in orange the dissemination and exploitation activities (WP6). WP1 spans the entire duration of the project. Similarly, WP6 starts at the beginning of the project for two reasons: (i) we plan for early dissemination through a website and social media, (ii) we will consider releasing the project artifacts as research and implementation progress along with the project.

In red, the technical WPs (WP2, WP3, and WP4) will start and proceed through the project in parallel. In green, the activities concerning the integration of the work packages and the validation of the proposed solutions last until the end of the project, starting from M7. In particular, the activities related to the integration (T5.1) will be carried out in the second year. We plan to combine the best features of automated test case generation and formal methods to provide an integrated solution to verify BSC. This activity will be conducted as part of T3.2, T4.2, and T5.2, starting from M19. Finally, we plan to periodically assess the designed methods and tools (i.e., T5.3) starting from M7. We highlight that during the integration and validation periods, the technical WPs (WP3, WP3, and WP4) will not stop but will be involved to address bug fixes, modifications, or feedback from integration and validation activities.

**ROLE AND CONTRIBUTION OF THE RESEARCH UNITS**

The project development will exploit the research background of the project participants. Their research records clearly show that they can provide complementary knowledge and expertise for developing the project and delivering the expected results. Thus, each unit will provide unique expertise and main role to address specific objectives of the project and will collaborate with the others, as shown in the following:

**UNISA**

*Expertise*: empirical software engineering, search-based software engineering, software quality, software defect prediction, mining software repositories, test case generation.
*Main Role*: (i) coordination, (ii) work package leader for WP2 to frame a fault model for BSC, (iii) experimentation of the attack models, (iv) experimental assessment, (v) dissemination.
*Other Contributions to Technical Packages*: (i) automated generation of test case scenarios in WP3, (ii) automated generation of testing oracles in WP3.

**UNITN**

*Expertise*: formal methods theory and tools, construction and analysis of formal models, stochastic and timed system models, exhaustive and statistical model checking.
*Main Role*: (i) work package leader for WP4 to design and implement the timed-stochastic models that formalize the network of BSC, (ii) combination of automated test case generation and formal methods, (iii) open-source release of the project artifacts.
*Other Contributions to Technical Packages*: (i) catalog of defects and vulnerability issues in WP2, (ii) automated generation of error test scenarios in WP3, (iii) experimentation of the attack models in WP4.

**UNIVR**

*Expertise*: test case generation, automated testing oracles, security testing, empirical studies, code hardening.
*Main Role*: (i) work package leader for WP3 to design and implement methods for the automated generation of test cases, (ii) integration, (iii) research showcase in industrial contexts.
*Other Contributions to Technical Packages*: (i) mining datasets of software defects, (ii) mutation seeding tool to inject faults in BSC in WP2, (iii) formal models of networks of BSC in WP3.

Each unit will provide its available hardware and infrastructural resources to carry out the project activity, such as servers and equipment (e.g., UNISA has acquired them on the Department of Excellence fundings). However, more blockchain-specific infrastructure could be needed to complete the activities. This infrastructure will be acquired using general funding.

---

## *4. Possible application potentialities and scientific and/or technological and/or social and/or economic impact*

Smartitude could significantly impact the technological, economic, social, and scientific points of view. In the following, we provide details for each aspect.

**TECHNOLOGICAL AND ECONOMIC IMPACT**

Decentralized finance (DeFi) is changing finance, having prominence from the summer of 2020 with yield farming and the rise of tokens promising attractive yields to crypto traders. According to DeFi Pulse, the total value of cryptocurrencies locked in DeFi protocols is over $78 billion in February 2022—a growth of 10x since May 2020. The number of users using DeFi applications is estimated at 4 million—a growth of over 40x in the last two years. Ethereum is the core blockchain powering DeFi applications. It is the second-largest cryptocurrency by market cap, with a volume higher than Visa (i.e., the second-largest payment processing company) in 2021. DeFi challenges traditional finance, replacing human trust with math-based trust, paperwork with smart contracts, legal enforcement with cryptographic enforcement, and third-party audit with open source code and public ledger. Developers can build new financial products featuring decentralized banking, decentralized money markets, and decentralized asset management. DeFi aims to be 10x better, faster, and cheaper than today's financial services.

Let us analyze payments performed using cryptocurrencies. When a consumer makes an online payment in centralized finance, various intermediaries are involved, increasing the transaction cost. In decentralized finance, the same transaction can be executed and validated by blockchains such as Ethereum, which form the base layer of the DeFi stack and act as single ground truth for all transactions within the network. When a transaction is broadcasted on the network, it is processed, cleared, and settled, eliminating the need for any intermediaries. Some cryptocurrencies (i.e., Stablecoins) were designed with fixed values to prevent their volatility, emerging as faster and cheaper alternatives for making domestic and cross-border payments globally. They reached a market cap of $170 billion.

In DeFi, paperwork is replaced by blockchain smart contracts. The source code of these digital agreements could be affected by software defects and vulnerabilities [Rek2022]. Trivial bugs could (and have ) had astonishing effects. For example, on May 5th, 2021, the Value DeFi protocol lost $10 million due to a hack because of a missing initialization line that allowed the exploiter to re-initialize the pool contract and act as its owner. The code was migrated from an old implementation of the Value DeFi Reserve Fund, which had the correct setting. Nevertheless, when merging the code, the line was not included.

Smartitude will provide a holistic fault model for issues occurring during BSC software development to provide developers with a catalog of possible issues that could affect their code. Furthermore, it will provide methods and tools to detect software defects and vulnerabilities like those exploited in Value DeFi. The tools will be extensible to help security experts and bug bounty hunters catch new defects, as incentivized by Binance Smart Chain with "Priority One," a $10 million fund for projects built on top of the BSC protocol [Red2021]. This initiative aims to "refine the lifecycle management of BSC users and decrease project exploits," showing the extent to which having methods and tools to detect defects and security vulnerabilities of BSC is essential for DeFi.

**SOCIAL IMPACT**

Pseudo-anonymity and decentralization provide a favorable environment for criminals [Cha2022]. Even though blockchain transactions are traceable—since most blockchains are public and every transaction is logged—several services and techniques can enhance anonymity and hinder law enforcement investigations. Among the issues related to a DeFi system, research effort is needed to secure crypto-wallets from theft via hacking. Such activities can be carried out in several ways. In some cases, hackers have exploited vulnerabilities in ledgers and stolen coins worth millions euros.

Scamming revenues are increasingly rising [Cha2022], particularly from rug pulls, a relatively new scam type where developers build apparently legitimate cryptocurrency projects before taking investors' money and disappearing.
Rug pulls are common in decentralized finance for two reasons. First, DeFi transaction volume has continuously grown since their introduction partially because of the extraordinary returns on decentralized tokens, which fueled many excited speculations on DeFi tokens.
At the same time, it is elementary for those with the right technical skills to create new DeFi tokens and list them on exchanges without any code audit. Code audits allow a third-party firm or listing exchange to analyze the BSC code behind new tokens or DeFi projects to provide a quality assurance certification. Rug pulls could have a lower impact if BSC were inspected through code audits. The testing and vulnerability assessment methods and tools provided by Smartitude will facilitate code audits to verify their quality and limit criminal activities.

**SCIENTIFIC IMPACT**

Several initiatives have demonstrated interest in BSC testing and vulnerability assessment in the last years. For example, the Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB) regularly includes discussions on these topics. The motivations for this workshop have increased in the past two editions given the clear outstanding interest of software communities, research communities, industry, finance, and media in Blockchain Oriented Software Engineering, as expressed by the increasing number of publications, start-ups, and companies exploiting this technology. Nonetheless, besides the growing interest, there is still a lack of automated testing methods and tools for Blockchain-oriented software.
The software engineering research community widely investigates software testing and software security, with considerable space in top-tier software engineering journals and conferences. Thus, it is easy to predict that Smartitude will contribute to providing further evidence about the usefulness of these methods and will obtain the attention of the Software Engineering research community.
Top-level publications and demonstrations will measure the scientific impact of the project, which will exploit the assets of the proposal from a scientific perspective: (i) the fault model for BSC, including a mutation seeding tool to inject faults in BSC, (ii) methods and tools to automatically generate test case for BSC, (iii) methods and tools for automated vulnerability and risk assessment of BSC.

**DISSEMINATION ACTIONS**

The methods and tools provided by Smartitude will be promoted in different communities:

- *Software Designers and Developers*. Smartitude will ease the testing and vulnerability assessment of software systems developed using BSC; therefore, we aim to inform this category of software designers and developers. *Specific dissemination actions*. We will participate in industrial and tutorial tracks of scientific and industrial events to provide demonstrations and hands-on sessions where the Smartitude solution will be shown in practice. Finally, we will employ Smartitude methods and tools in the context of the ICE Laboratory of the University of Verona, as highlighted in our case study.

- *Academia and Researchers* are important because they can contribute to improving the tools provided by Smartitude via a sustainable open-source process. The dissemination activities will target scientific journals, conferences, and other academic/research events dedicated to software testing, security, and formal methods. *Specific dissemination actions:* The results provided by Smartitude can be of interest to different communities related to software verification and validation. These communities are involved and maintain many international journals and conferences. Some relevant scientific journals and conferences that Smartitude will target are IEEE Trans. on Softw. Eng. (TSE), ACM Trans. Softw. Eng. and Methodology

(TOSEM), Empirical Software Engineering (Springer), Sw. Tools for Tech. Transfer (STTT), Int. Conf. on Soft. Eng. (ICSE), ACM Europ. Sw. Eng. Conf./Symp. on the Found. Sw. Eng. (ESEC/FSE), Int. Conf. on Automated Soft. Eng. (ASE), ACM SIGSOFT Int. Symp. on Sw. Testing and Analysis (ISSTA), Int. Conf. on Tools and Alg. for the Constr. and Anal. of Syst. (TACAS), Int. Conf. on Sw. Testing, Verification and Validation (ICST), IEEE Computer Security Found. Symp. (CSF). Furthermore, the project partners will organize a workshop to foster the solutions delivered and increase their impact.

The partners have the capabilities and networks needed to address the markets targeted by the project results, as shown by their academic records in Sec. B.2.

## *5. Financial aspects: costs and funding for each research unit*

| nº | | Total cost (euro) | Co-funding (item A.1) (euro) | MUR funding (other items) (euro) |
|---|---|---|---|---|
| 1. | DI NUCCI Dario | 103.200 | 19.500 | 83.700 |
| 2. | BUDDE Carlos Esteban | 101.838 | 20.322 | 81.516 |
| 3. | CECCATO Mariano | 84.000 | 0 | 84.000 |
| | **Total** | **289.038** | **39.822** | **249.216** |

## *6. Bibliography*

[Aba2021] Abate, A. et al. (2021). ARCH-COMP21 Category Report: Stochastic Models. ARCH21, vol. 80, 55-89.

[Alm2018] Almukaynizi, M., Paliath, V., Shah, M., Shah, M. & Shakarian, P. (2018). Finding Cryptocurrency Attack Indicators Using Temporal Logic and Darkweb Data. IEEE ISI, 91-93.

[And2016] Anderson, L., Holz, R., Ponomarev, A., Rimba, P., & Weber, I. (2016). New kids on the block: an analysis of modern blockchains. arXiv preprint arXiv:1606.06530.

[Atz2017] Atzei, N., Bartoletti, M., & Cimoli, T. (2017). A survey of attacks on ethereum smart contracts (sok). POST, 164-186. Springer, Berlin, Heidelberg.

[Bud2021] Budde, C.E. et al. (2021). On Correctness, Precision, and Performance in Quantitative Verification. In ISoLA 2020, LNCS vol. 12479. Springer.

[Del2015] Delmolino, K., Arnett, M., Kosba, A., Miller, A., & Shi, E. (2015). Lab: Step by Step towards Programming a Safe Smart Contract.

[Fra2014] Fraser, G., & Arcuri, A. (2014). A large-scale evaluation of automated unit test generation using evosuite. ACM Transactions on Software Engineering and Methodology (TOSEM), 24(2), 1-42.

[Gar2019] Garousi, V., Felderer, M., & Mäntylä, M. V. (2019). Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. IST, 106, 101-121.

[Gha2020] Ghaleb, A., & Pattabiraman, K. (2020). How effective are smart contract analysis tools? evaluating smart contract static analysis tools using bug injection. 29th ACM SIGSOFT ISSTA, 415-427.

[Gre2019] Greenberg, A. (2019). A 'Blockchain Bandit' Is Guessing Private Keys and Scoring Millions. https://www.wired.com/story/blockchain-bandit-ethereum-weak-private-keys/

[Gri2018] Grishchenko, I., Maffei, M. & Schneidewind, C. (2018). Foundations and Tools for the Static Analysis of Ethereum Smart Contracts. In CAV 2018. LNCS vol. 10981. Springer.

[Hil2018] Hildenbrandt et al. (2018). KEVM: A Complete Formal Semantics of the Ethereum Virtual Machine. IEEE 31st CFS, 204-217.

[Jia2018] Jiang, B., Liu, Y., & Chan, W. K. (2018). Contractfuzzer: Fuzzing smart contracts for vulnerability detection. 2018 33rd IEEE/ACM ASE, 259-269.

[Kal2018] Kalra, S., Goel, S., Dhawan, M. & Sharma, S. (2018). ZEUS: Analyzing Safety of Smart Contracts. NDSS 2018.

[Kit2007] Kitchenham, B., & Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering.

[Lid2010] Lidwell, W., Holden, K., & Butler, J. (2010). Universal principles of design, revised and updated: 125 ways to enhance usability, influence perception, increase appeal, make better design decisions, and teach through design. Rockport Pub.

[Liu2019] Liu, H., Yang, Z., Jiang, Y., Zhao, W. & Sun, J. (2019). Enabling Clone Detection For Ethereum Via Smart Contract Birthmarks. IEEE/ACM 27th ICPC, 105-115.

[Liu2020] Liu, Y., Li, Y., Lin, S. W., & Yan, Q. (2020). ModCon: A model-based testing platform for smart contracts. 28th ACM ESEC/FSE, 1601-1605.

[Liu2022] Liu Y., Xu J. & Cui B. (2022) Smart Contract Vulnerability Detection Based on Symbolic Execution Technology. CNCERT 2021: Cyber Security. CCIS vol. 1506. Springer.

[Luu2016] Luu, L., Chu, D. H., Olickel, H., Saxena, P., & Hobor, A. (2016). Making smart contracts smarter. 2016 ACM SIGSAC CCS, 254-269.

[Mue2021] Mueller, B. (2021). Wolf Game Herdsman's report. https://muellerberndt.medium.com/wolf-game-herdsmans-report-5802c9d477bd.

[Mun2017] Munaiah, N., Kroh, S., Cabrey, C., & Nagappan, M. (2017). Curating github for engineered software projects. EMSE, 22(6), 3219-3253.

[Nak2008] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Decentralized Business Review, 21260.

[Rea2016] Alex, R. (2016). Code Coverage for Solidity. https://blog.colony.io/code-coverage-for-solidity-eecfa88668c2/

[Red2021] Jamie Redman (2021). Binance Smart Chain Creates a $10 Million Bug Bounty Fund to Tighten Protocol Security. https://news.bitcoin.com/binance-smart-chain-creates-a-10-million-bug-bounty-fund-to-tighten-protocol-security/

[Rek2022] Rekt (2022). Rekt Leaderboard. https://rekt.news/leaderboard/

[Tor2018] Torres, C.F., Schütte, J. & State, R. (2018). Osiris: Hunting for Integer Bugs in Ethereum Smart Contracts. 34th ACM ACSAC, 664-676.

[Vig2020] Viglianisi, E., Ceccato, M., & Tonella, P. (2020). A federated society of bots for smart contract testing. JSS, 168, 110647.

[Wan2021] Wang, Y., He, J., Zhu, N., Yi, Y., Zhang,Q., Song, H. & Xue, R. (2021). Security enhancement technologies for smart contracts in the blockchain: A survey. Trans Emerging Tel Tech.; 32(12):e4341.

[Woo2014] Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper, 151(2014), 1-32.

[Zha2019] Zhang, P., Xiao, F., & Luo, X. (2019). Soliditycheck: Quickly detecting smart contract problems through regular expressions. arXiv preprint arXiv:1911.09425.

[Zha2020a] Zhang, Y., Ma, S., Li, J., Li, K., Nepal, S., & Gu, D. (2020). Smartshield: Automatic smart contract protection made easy. 2020 IEEE 27th SANER, 23-34.

[Zha2020b] Zhang, P., Yu, J., & Ji, S. (2020, June). ADF-GA: data flow criterion based test case generation for ethereum smart contracts. IEEE/ACM 42nd ICSE Workshops, 754-761.