NtaTrust

Smart contracts and blockchain protocols are written in expressive programming languages such as Solidity, C/C++, Java and Go. These languages are Turing-complete, which intuitively means that they are very expressive and flexible. The price to pay is that almost all interesting problems are undecidable. Therefore, it is impossible to develop a single method to ensure the quality of blockchain ecosystems. We thus provide a holistic approach that combines the strength of multiple system analysis methods.

NtaTrust is a quality assurance platform for blockchain ecosystems that can falsify or verify performance properties and functional properties (correctness and security properties). The platform builds upon five core techniques, including **dynamic analysis**, **fuzz testing**, **symbolic execution, model checking** and **static analysis**, that have synergistic and complementary strengths.

Dynamic analysis engine *NtaDyamic* goes beyond simply executing the program under a given input by conducting additional analyses of the program behavior to detect vulnerabilities. It has the capability of deriving a predictive model that captures additional program behaviors that have not be tested. Using constraint solving, *NtaDyamic* is able to detect vulnerabilities hidden in the predictive model and thus avoid the additional overhead of having to explore these program behaviors.

Fuzz testing engine *NtaFuzzy* can reveal many serious defects overlooked by blockchain developers with a high benefit-to-cost ratio. It accomplishes this by generating a massive amount of input data (called seeds) for the test subject. *NtaFuzzy* leverages advanced static analysis and constraint-solving techniques to drastically improve the quality of the seeds.

Symbolic execution engine *NtaSymbolic* assumes symbolic values for inputs rather than concrete values as in normal executions. It has the capability to systematically explore the program behavior and uncover corner cases. Specifically, *NtaSymbolic* leverages customized symbolic execution algorithms to enumerate path that are critical to the validity of functional and performance properties, including those due to *fallback* functions.

Model checking engine *NtaFormal* proves that a property holds under all possible execution conditions. When a functional or performance property holds, it is able to generate a mathematical proof. *NtaFormal* consists of three innovative components: a model builder to capture the precise semantics of a smart contract or blockchain protocol, a framework to specify a wide range of functional and performance properties, and a set of algorithms to efficiently verify the property in the model, e.g., using fully automated abstraction and refinement techniques.

Static analysis engine *NtaStatic* examines source code without executing the program. The process provides an understanding of the code structure and can help to ensure that the code adheres to industry standards. *NtaStatic* exploits algorithms specifically designed for smart contracts and protocols to evaluate and examine various aspects of the source code.

Both *NtaDynamic* and *NtaFuzzy* are computationally efficient for detecting. While *NtaDynamic* predicts erroneous behaviors under a fixed input, *NtaFuzzy* varies inputs to monitor more program behaviors. *NtaSymbolic* exploits a more comprehensive and systematic technique for exploring program paths and uncovers subtle vulnerabilities. In these three cases, the explored behavior constitutes a sound under-approximation of the actual behavior. *NtaStatic* and *NtaFormal*, on the other hand, focus on sound over-approximations of the program behavior. Therefore, inside NtaTrust, *NtaStatic* and *NtaFormal* are geared toward verifying properties, whereas *NtaDynamic*, *NtaDynamic* and *NtaSymbolic* are geared toward detecting property violations.