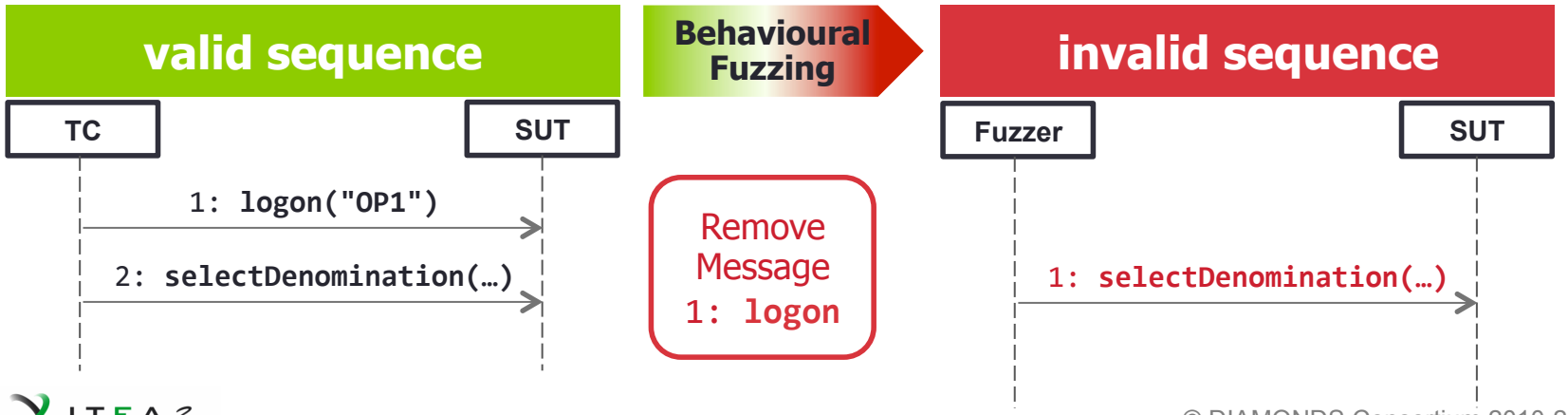
Development and Industrial Application of Multi-Domain Security Testing Technologies

Innovation Sheet
Model-Based Behavioural Fuzzing

ITEA 2
INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

Model-Based Behavioural Fuzzing
Description

- Test cases are generated by **fuzzing one or more valid sequences.**
- This concrete fuzzing of behaviour is realized by changing the order and appearance of messages in two ways:
  - **By rearranging messages directly**. This enables straight-lined sequences to be fuzzed. Fuzzing operators are for example remove, move or repeat a message.
  - **By utilising control structures of UML 2.x sequence diagrams**, such as combined fragments, guards, constraints and invariants. This allows more sophisticated behavioural fuzzing that avoids less efficient random fuzzing.
- By applying one ore more fuzzing operators to a valid sequence**, invalid sequences (= behavioural fuzzing test cases)** are generated.



ITEA 2
INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

Model-Based Behavioural Fuzzing
State of the art

- Behavioural fuzzing is rarely elaborated:
  - First mentioned by Kaksonen et al. during the PROTOS project (2001):
    "*In fault injection [i.e. fuzzing] mutations can be applied to the syntax of individual protocol messages as well as the order and type of messages exchanged.*"

    A. Takanen, J. DeMott and C. Miller, "Software Security Assessment through Specification Mutations and Fault Injection". In: Communications and Multimedia Security Issues of the New Century, Series: IFIP Advances in Information and Communication Technology, Vol. 64, Steinmetz, Ralf; Dittmann, Jana; Steinebach, Martin (Eds.), 2001
  - Becker et al. (2010) applied behavioural fuzzing to state machines by inserting, repeating and dropping messages based on the behavioural model.

    S. Becker, H. Abdelnur, R. State, T. Engel: An Autonomic Testing Framework for IPv6 Configuration Protocols. In: Mechanisms for Autonomous Management of Networks and Services. Lecture Notes in Computer Science. Stiller, Burkhard and De Turck, Filip (Eds.). Springer Berlin/Heidelberg. 2010.
  - SNOOZE, a tool for building for network protocol fuzzers uses state machines for protocol description and provides primitives for retrieving invalid messages depending on a state and thus, enables developing behavioural fuzzers.

    Banks, Greg and Cova, Marco and Felmetsger, Viktoria and Almeroth, Kevin and Kemmerer, Richard and Vigna, Giovanni. SNOOZE: Toward a Stateful NetwOrk prOtocol fuzZEr. In: Information Security. Lecture Notes in Computer Science. Katsikas, Sokratis and López, Javier and Backes, Michael and Gritzalis, Stefanos and Preneel, Bart (Eds.), 2006
  - Kitagawa et al. showed the effectiveness of behavioural fuzzing by finding vulnerabilities, e.g. in Apache web server, that could not be found by data fuzzing, but did not describe their test generation method.

    T. Kitagawa, M. Hanaoka, K. Kono: AspFuzz: A state-aware protocol fuzzer based on application-layer protocols. In: Symposium on Computers and Communications (ISCC'10), pp. 202–208. IEEE (2010)

Model-Based Behavioural Fuzzing
Advances beyond the state of the art

- Behavioural Fuzzing is extended to UML sequence diagrams by providing a rich set of behavioural fuzzing operators explicitly made for sequence diagrams. These fuzzing operators take advantage from the structures of sequence diagrams, e.g. combined fragments. A concrete method for generating test bases by applying fuzzing operators was developed. The approach enables reusing of functional test cases for security testing.

  [Deliverable D2.WP2, Section 2.8]

- A method how test case generation may benefit from risk assessment results by augmenting the model with security-related annotations was presented. This helps to focus on certain security aspects while reducing the number of test cases.

  [Deliverable D3.WP2, Section 2.3]

Model-Based Behavioural Fuzzing
Exploitation and application to case studies

- Model-based behavioural fuzzing was applied to Giesecke & Devrient case study.
- A prototype of a test case generator was developed.