Evaluation of the CORAL Approach for Risk-driven Security Testing based on an industrial Case Study

Gencer Erdogan, Ketil Stølen, Jan Øyvind Aagedal

SINTEF

UiO : University of Oslo

Background and Objective

- Security testers face the problem of determining tests that are most likely to reveal severe security vulnerabilities.
- We have developed a method for risk-driven security testing supported by a domain-specific modeling language referred to as CORAL.
- CORAL aims to help security testers to select and design test cases based on the available risk picture.
- Objective: Evaluate to what extent CORAL helps security testers in selecting and designing test cases.

The CORAL Language

- The CORAL language is based on UML interactions.

Messages

Node type	Notation
General message	signature →
New message: A message initiated by a threat	signature ▷→
Altered message: A message in the system under test altered by a threat	signature ▷→
Deleted message: A message in the system under test deleted by a threat	signature ⊠→
Unwanted incident message: Indicates that an asset is harmed	signature ✸→

Risk-measure annotations

Node type	Notation
Frequency: Represents either the frequency of the transmission or the reception of a message	frequency : time unit
Conditional ratio: Represents the ratio by which a message is received, given that it is transmitted	conditional ratio
Consequence: Represents the consequence an unwanted incident has on an asset	consequence

Lifelines

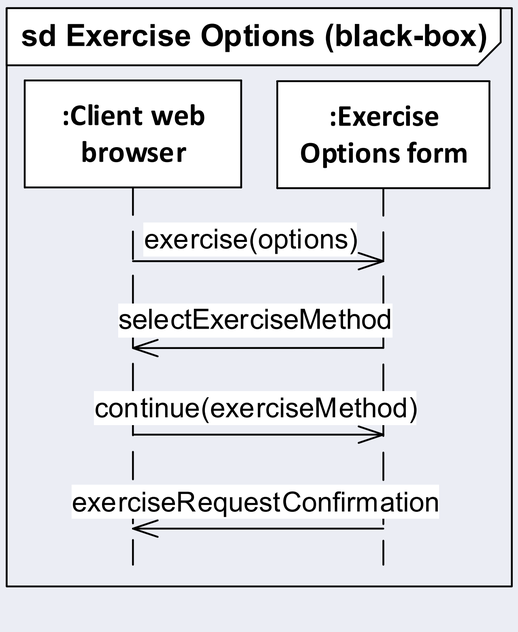
Node type	Notation
General lifeline	Name
Deliberate threat lifeline: A human threat that has malicious intents	Name
Accidental threat lifeline: A human threat that does not have malicious intents	Name
Non-human threat lifeline: A threat that may be anything else except a human	Name
Asset lifeline: Represents a security asset we want to protect	Name

Overview of Case Study

- The case study was carried out in three phases which correspond to the overall steps in CORAL: Test planning, security risk assessment, and security testing.
- The system under test was a web-based application providing services related to equity-based compensation plans.

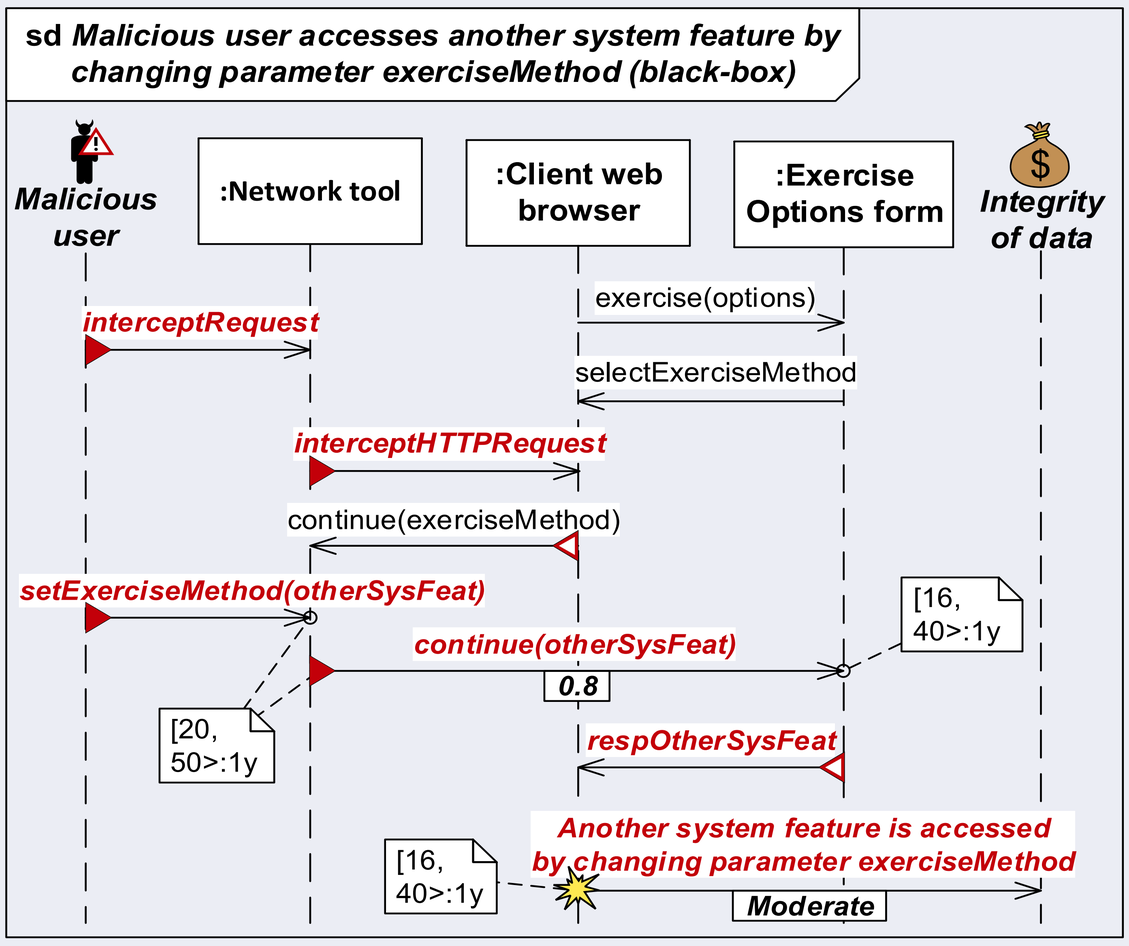
Test planning (phase 1)

1. Prepare the system model (the figure on the right is a fragment of the system model in the case study).
2. Identify security assets to protect.
3. Define frequency and consequence scales.
4. Define the risk evaluation matrix.



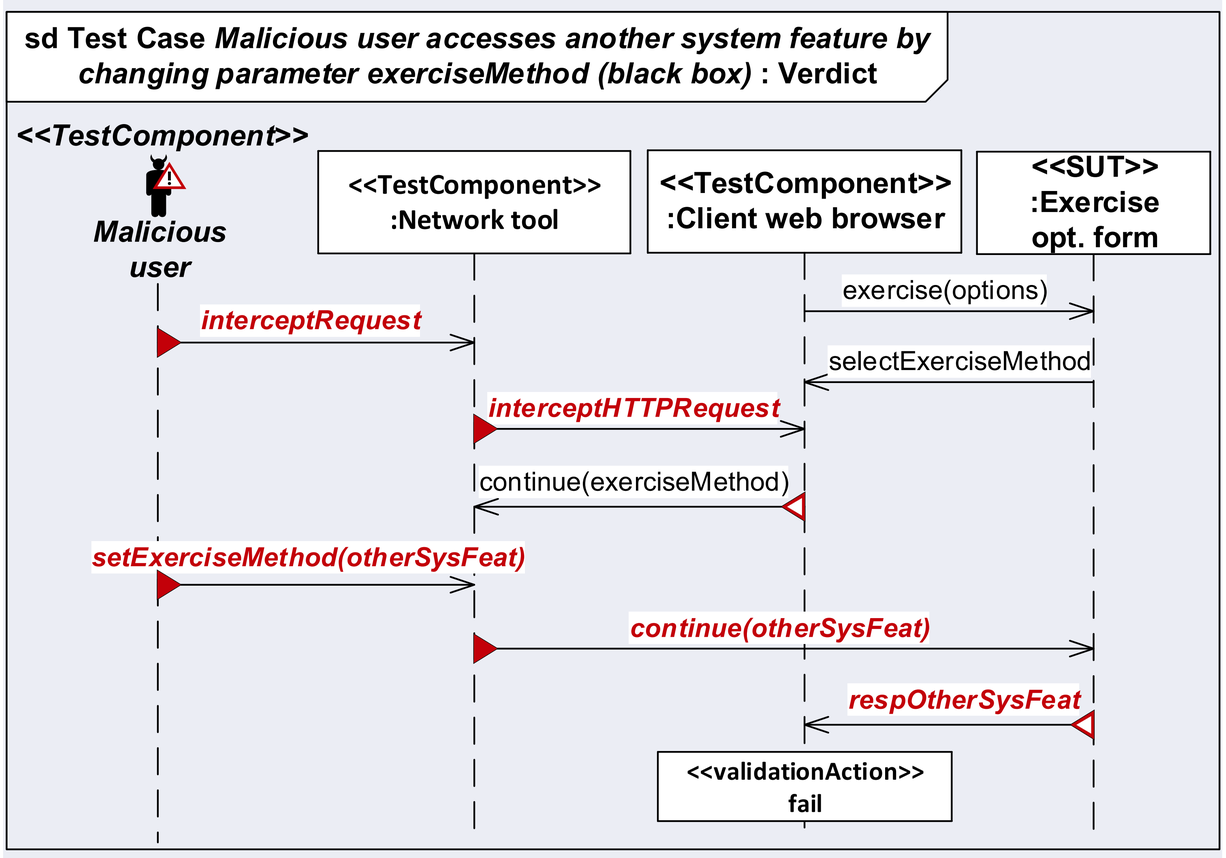
Security risk assessment (phase 2)

- Identify, estimate, and evaluate risks, and select risks to test.



Security testing (phase 3)

- Design security tests based on selected risks and execute tests.



Results and Discussion

- The case study results indicate that CORAL is effective in terms of producing valid risk models. This is backed up by two observations. First, we identified in total 21 risks, and 11 of these risks were considered as severe, while the remaining 10 risks were considered as low risks. By testing these 11 risks we identified 11 vulnerabilities, while by testing the remaining 10 risks we identified only 2 vulnerabilities. Second, we identified all relevant security risks compared to previous penetration tests. In addition, we identified five new security risks and did not leave out any risks of relevance for the features considered.
- The CORAL approach seems to work equally well for black-box and white-box testing.
- Finally, one of the most important findings we did in the case study is that the CORAL approach is very useful for identifying security test cases. We used all threat scenarios identified in the case study for the purpose of security test case design and execution.