25 – Assurance Cases and Confidence
Charles B. Weinstock
November 17, 2016

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213
Security Assurance Cases

An assurance case links a claim to evidence supporting that claim via a structured argument.

• E.g., The system is not susceptible to sql injection

Eliminative arguments build the argument by positing and then eliminating (by argument) counterexamples.

The more counterexamples that can be eliminated, the more confidence we have in the claim.

The existence of counterexamples that cannot be eliminated leads to doubt that the claim has been met.
The Problem

Given the evidence, how confident should we be in the claim $C_1$? Why?

What does it mean to have confidence in the claim?

What could be done to improve confidence? Why?

- $C_1$: The system is secure
- $C_2$: Vulnerability 1 has been eliminated
- $C_3$: Vulnerability 2 has been eliminated

Evidence:
- $E_{v1}$
- $E_{v2}$
- $E_{v3}$
The Basis for Confidence in a Claim

A classic philosophical problem:
• Justify belief in a hypothesis

Use Induction
• **Enumerative**: Support increases as **confirming instances** are found

Using past experience as the basis for predicting future behavior
The Basis for Confidence in a Claim

A classic philosophical problem:
• Justify belief in a hypothesis

Use Induction
• **Eliminative**: Support increases as *reasons for doubt* are eliminated
  - Switch not connected to light
  - No power
  - Dead light bulb
The Problem

How confident in C1? Why? (Number of uneliminated doubts)
What does it mean to have confidence? (Lack of doubt)
What could be done to improve confidence? Why? (Elim. more doubts)

- **C1**: The system is secure
  - **C2**: Vulnerability 1 has been eliminated
    - Ev1: evidence
  - **C3**: Vulnerability 2 has been eliminated
    - Ev2: evidence
    - Ev3: evidence
A Small Example

**Rebutting defeaters (R) attack claim validity**

**Cx2.1a**
A parameter is unrestricted if it can cause an unintended modification to the SQL query when used.

**R2.1**
Unless there is unrestricted user input to a query.

**Ev3.2**
Evidence showing that all user input is properly sanitized before being used in an SQL query.

**IR2.2**
If user input is properly restricted, then the system is safe from SQL injection.

**UC3.3**
Unless there is another way SQL injection could occur.

**Inference rule (IR) for validating a claim**

**Undermining defeaters (UM) attack evidence validity**

**UM4.1**
But the evidence is based upon faulty sanitation rules.

**UM4.2**
But the evidence is not for the system under consideration.

**IR4.3**
If these reasons are not true, then the evidence is valid.

**C1.1**
The system is secure from SQL injection.

**IR3.1**
If all user input is sanitized then there is no unrestricted user input to a query.

**Ev3.2**
Evidence showing that all user input is properly sanitized before being used in an SQL query.

**IR2.2**
If user input is properly restricted, then the system is safe from SQL injection.

**UC3.3**
Unless there is another way SQL injection could occur.

**Undercutting defeaters (UC) attack rule sufficiency**

**Ev3.2**
Evidence showing that all user input is properly sanitized before being used in an SQL query.

**IR2.2**
If user input is properly restricted, then the system is safe from SQL injection.

**UC3.3**
Unless there is another way SQL injection could occur.

**IR4.3**
If these reasons are not true, then the evidence is valid.
Security benefit for delivered software

Documenting reasons for (dis)belief (in a security claim) allows for more effective review

Stimulates search for holes in the argument
  • Residual doubts are highlighted and can be addressed (if thought necessary)

If there is a security violation related to a claim, having the assurance case makes it possible to determine where flaws exist in the case and probably in the design or implementation
Implied Requirements for Design / Development / Evaluation

The designers, developers, and evaluators have to be aware of common vulnerabilities and how they are combated (to build/evaluate the case.)

Once an argument is developed (e.g., regarding sql injection) it can be reused elsewhere as long as the required evidence still applies.

The case evolves with steps in the life cycle
Resources required of the developing organization

Training in the use of assurance cases and eliminative argumentation

Tooling

• Special purpose (e.g., ASCE)
• General purpose (e.g., Mindmanager, Excel)
• Text files
Method of Evaluation

Reviewer develops an evaluation case, i.e., a case whose claim is about the compliance of the item with the building code.

Reviewers need training and tools as for the developer.
Evidence of Effectiveness

Assurance cases have been effective in the safety domain

• Aviation
• Nuclear
• Rail
• Medical devices
• ….

The effectiveness of eliminative argumentation as a confidence evaluation method has yet to be demonstrated in practice
Closing Thoughts

Most development organizations are already creating much of the required evidence.

• The assurance case developed by using eliminative argumentation adds information that links the evidence to the desired claim.

An assurance case based on eliminative argumentation would be a good means of evaluating the proposed building code itself!