An Anti-Pattern for Misuse Cases

Mohammad Torabi Dashti
mohammad.torabi@inf.ethz.ch

Saša Radomirović
s.radomirovic@dundee.ac.uk

Contribution

- Presentation of orphan misuse cases anti-pattern: Elicitation of misuse cases that do not include any use case.

- Demonstration of negative consequences of orphan misuse cases.

- Guidance on how to avoid orphan misuse cases.

Context: Threat Modelling

- Structured approaches to elicit threats to a system
- Examples:
  - STRIDE: Elicit threats by focusing on categories
  - Attack Trees: Elicit threats by iterative refinement
- Different strategies, common problem:
  - Open World Assumption: You are never done, there are always threats that have not been considered

STRIDE

Elicit threats by categories

Example threats to online banking web application:

Spoofing (Authenticity): Impersonate customer (XSS, SQL inj)

Tampering (Integrity): Forge transactions (CSRF)

Repudiation (Non-repudiation): Delete audit logs (command inj)

Information disclosure (Confidentiality): View balances (SQL inj)

Denial of service (Availability): Flood server with TCP packets

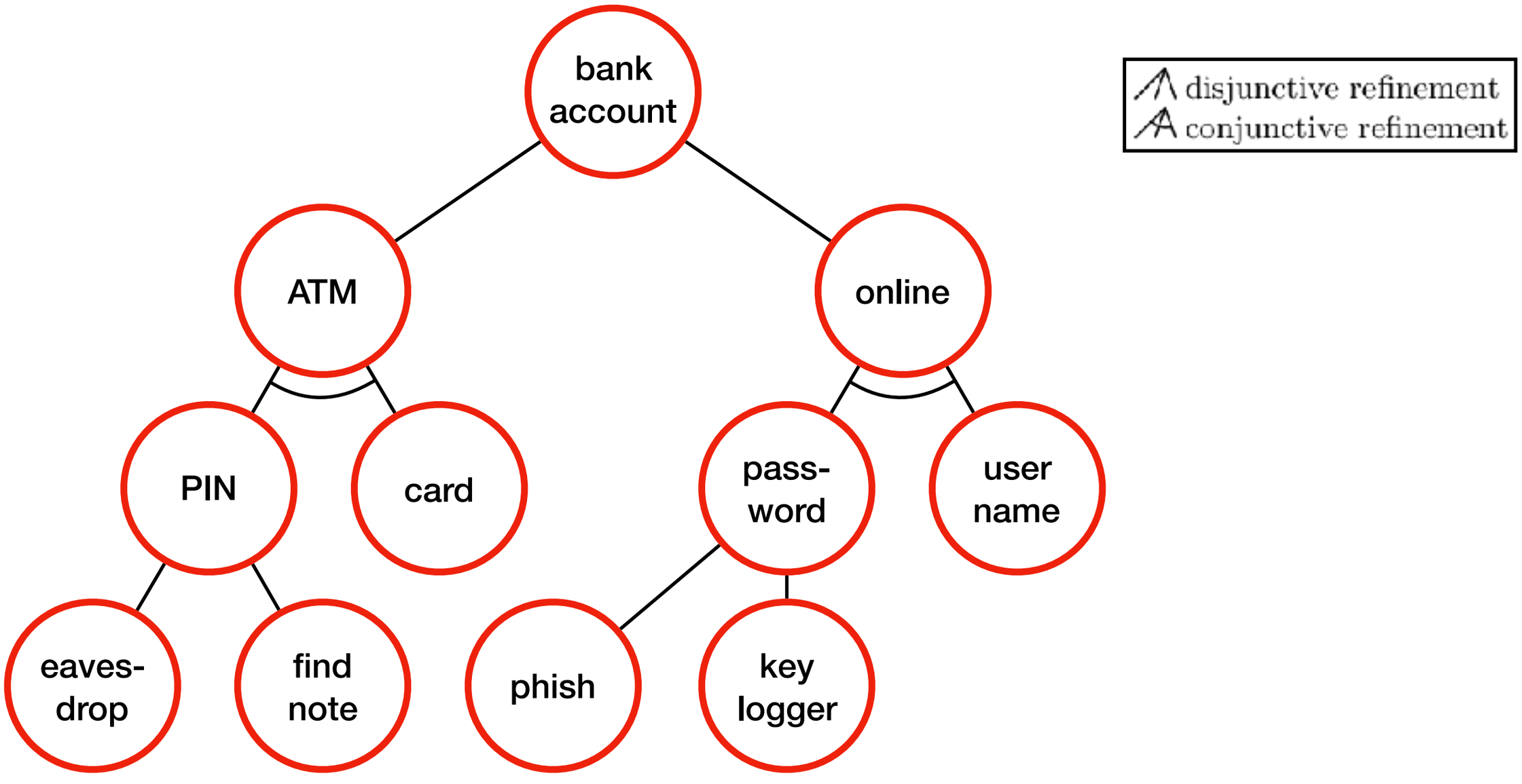
Elevation of privilege (Authorization): **Blackmail system admin?**

Attack Trees

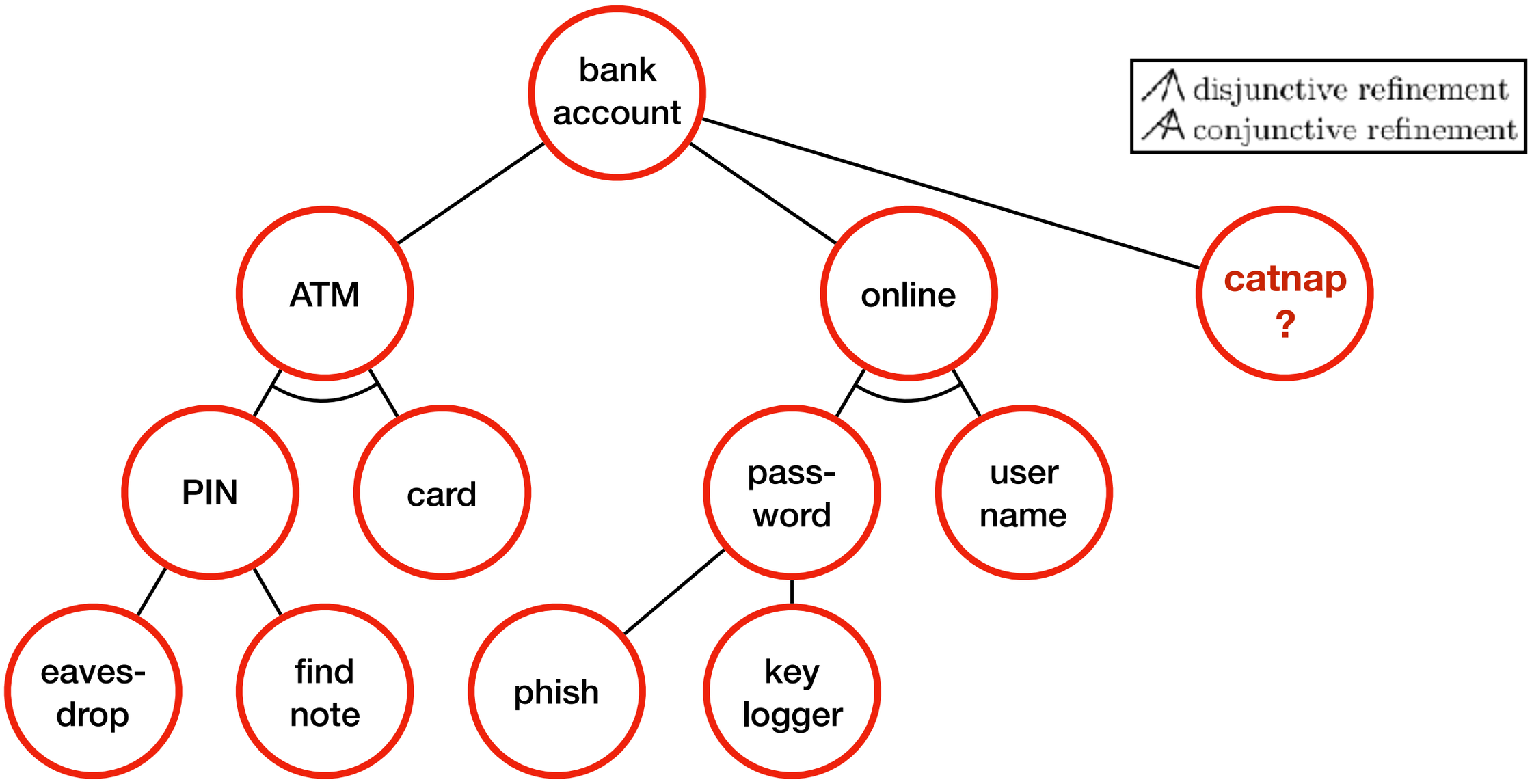
Elicit threats by iterative refinement

Attacker’s goal: Steal money from a bank account

Attack Trees

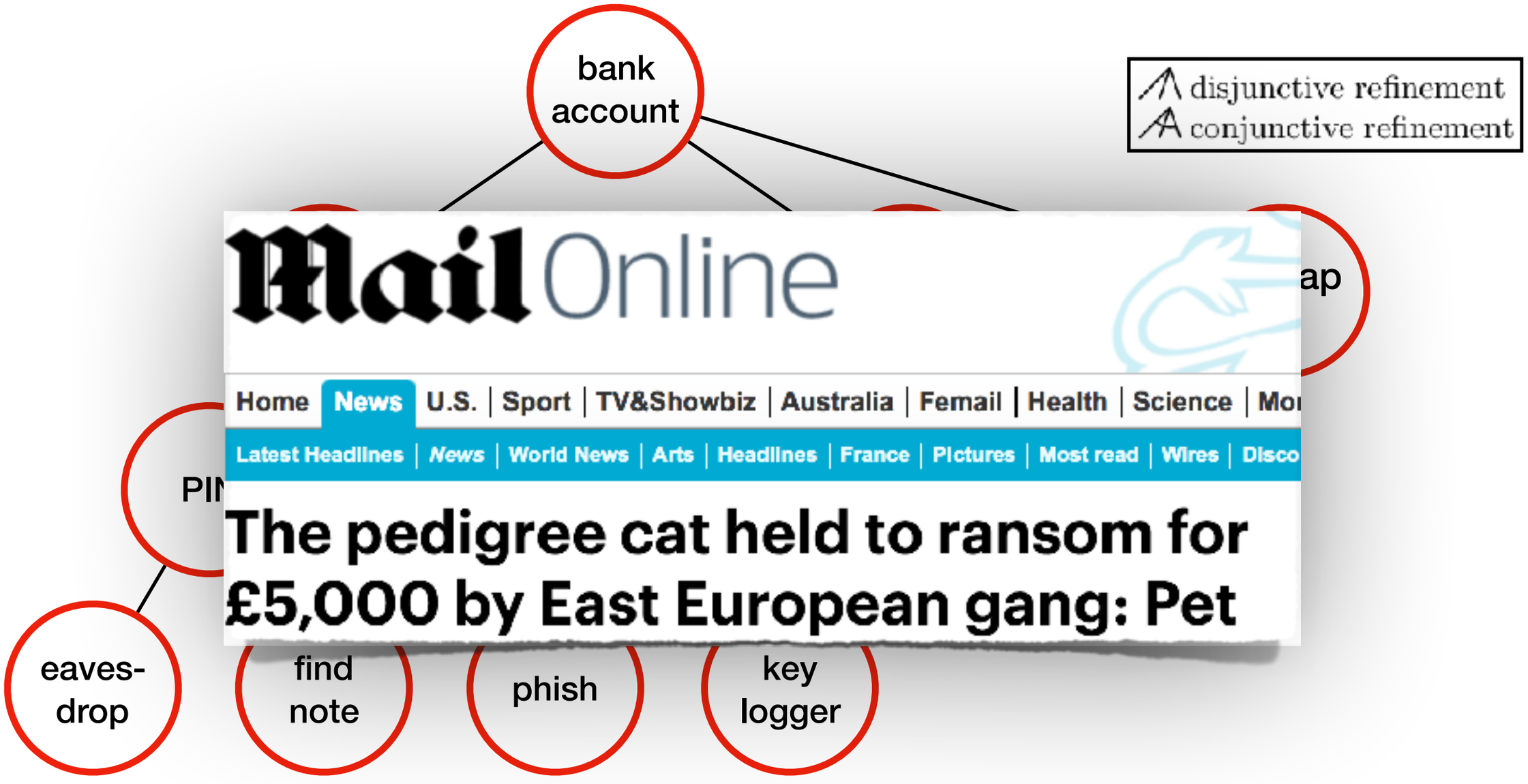


Attack Trees



What about holding the bank account owner's cat to ransom?

Attack Trees



What about holding the bank account owner's cat to ransom?

Threat Modelling

Problems:

- Open World Assumption: You are never done, there are always threats that have not been considered.
- No guidelines:
  - Which threats should be included in the model?
  - What is the right level of detail/abstraction?

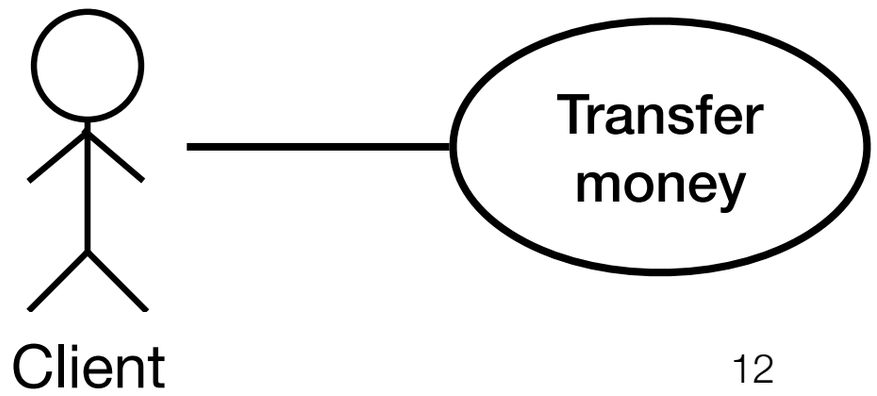
Symptom:

- Analysis Paralysis
  - Mitigation: Use cases provide a cognitive catalyst by limiting search space.

Use Cases

- Used in software engineering for requirements elicitation.
- Requirements expressed informally as narratives involving
  - actors (interacting with the system) and
  - use cases (functionality provided by the system)

- **Online Banking Example**: *The client can transfer money to an external account by providing the recipient's account number and the amount. The system verifies that there are sufficient funds available, shows transaction details and provides the options to confirm or cancel the transfer. […]*

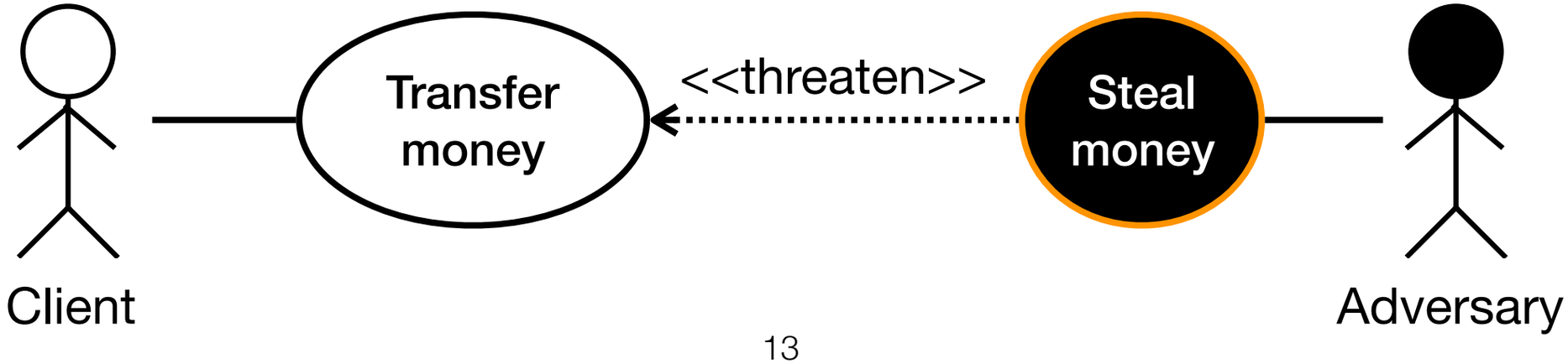


Misuse Cases

- **Elicit security requirements** from use cases and security objectives.
- This can be done using STRIDE, Attack Trees, …

Example:

- Security objective: *Integrity of Client's assets.*
- Misuse case: *The adversary uses the functionality to transfer money to his own account without the Client's approval.*

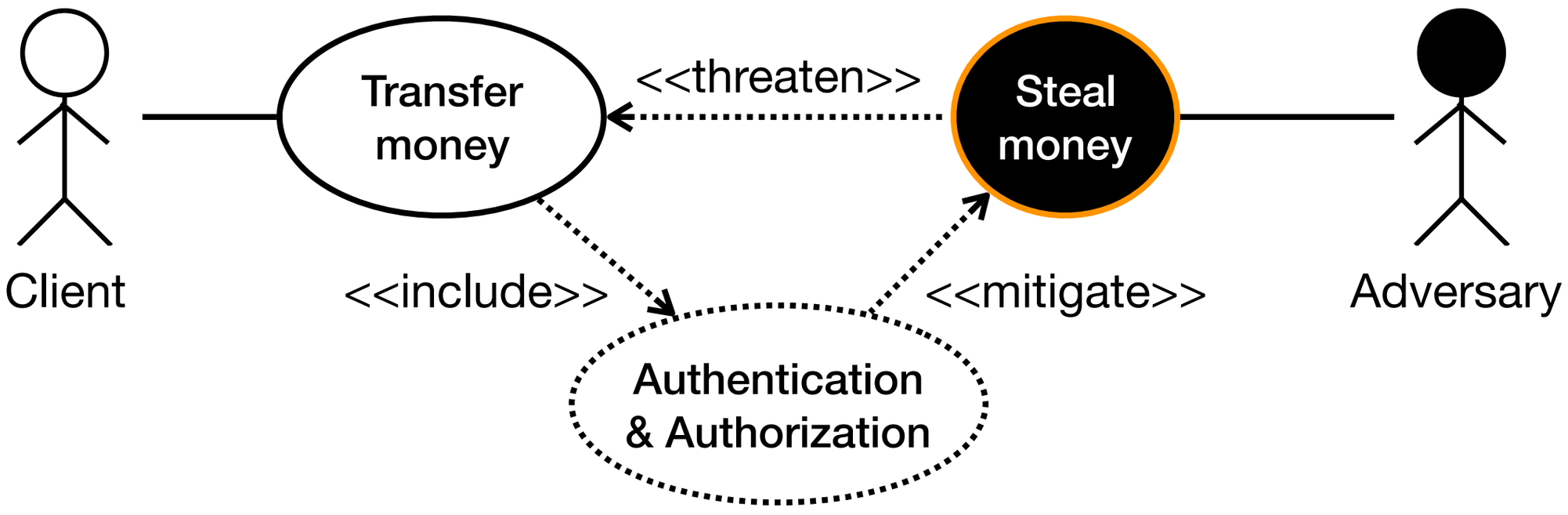


Mitigation

- Misuse cases are mitigated by eliciting new functional requirements (security use cases).

Example:

- Mitigation: *Client is authenticated to verify that the transfer is initiated by the Client.*

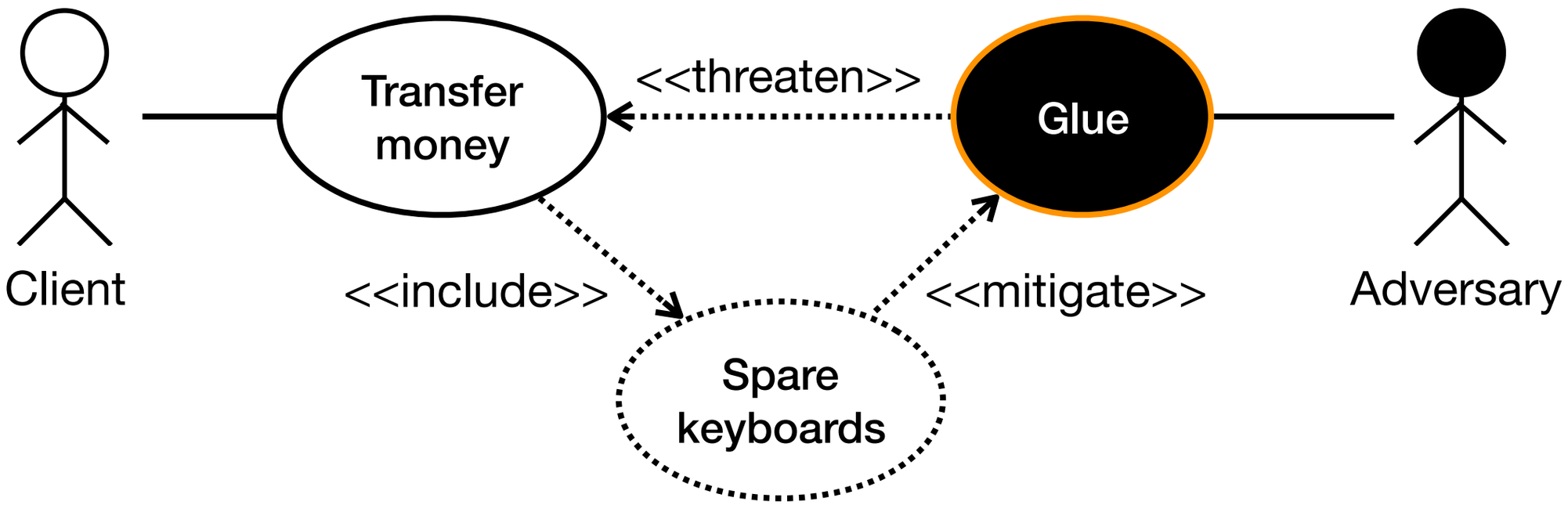


Problem

- You can write *anything* into these bubbles.

Example:

- Misuse: *Immobilize the <return> key with superglue.*
- Mitigation: *Keep spare keyboards at hand.*

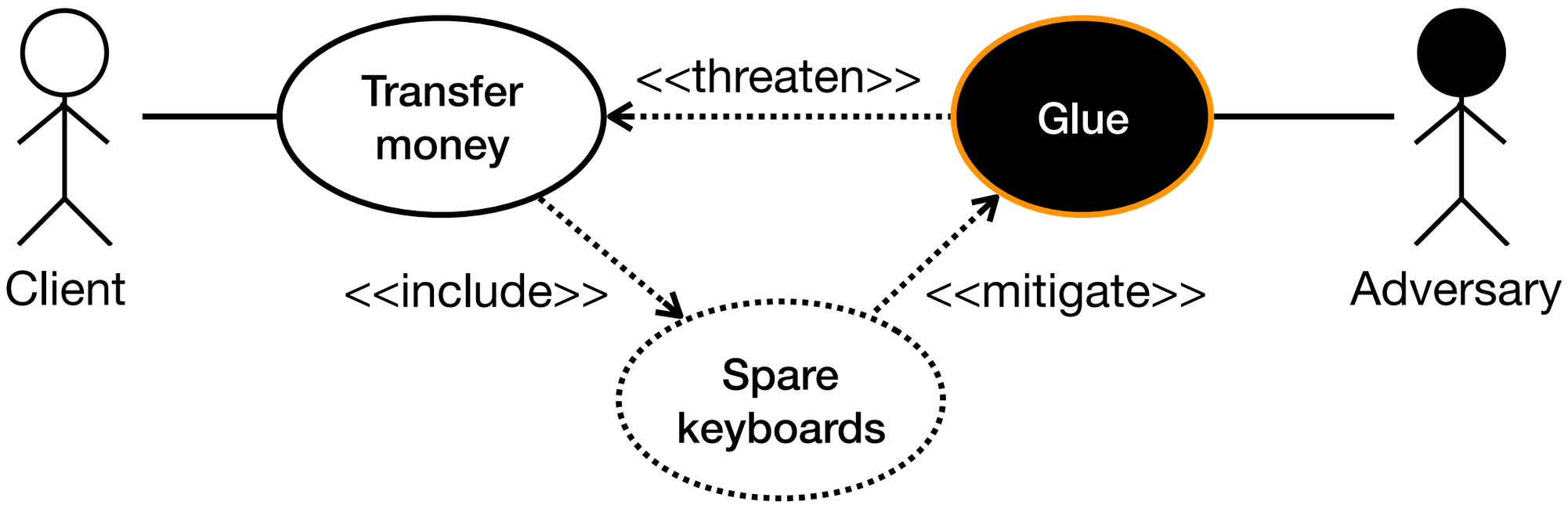


Note: sabotage of use case ≠ misuse

Orphan Misuse Cases

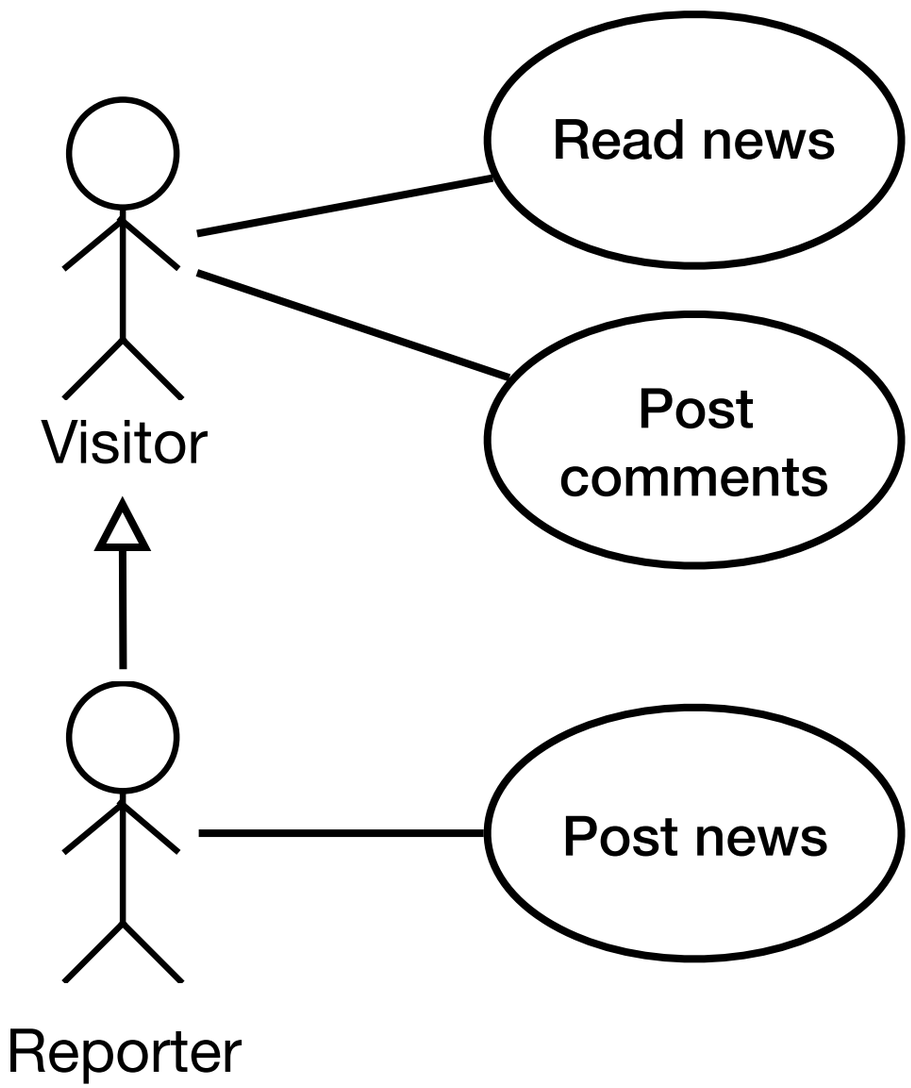
Use cases that do **not include** a use case, e.g., because they

- are implementation dependent,
- threaten almost all use cases,
- have vacuous mitigations.



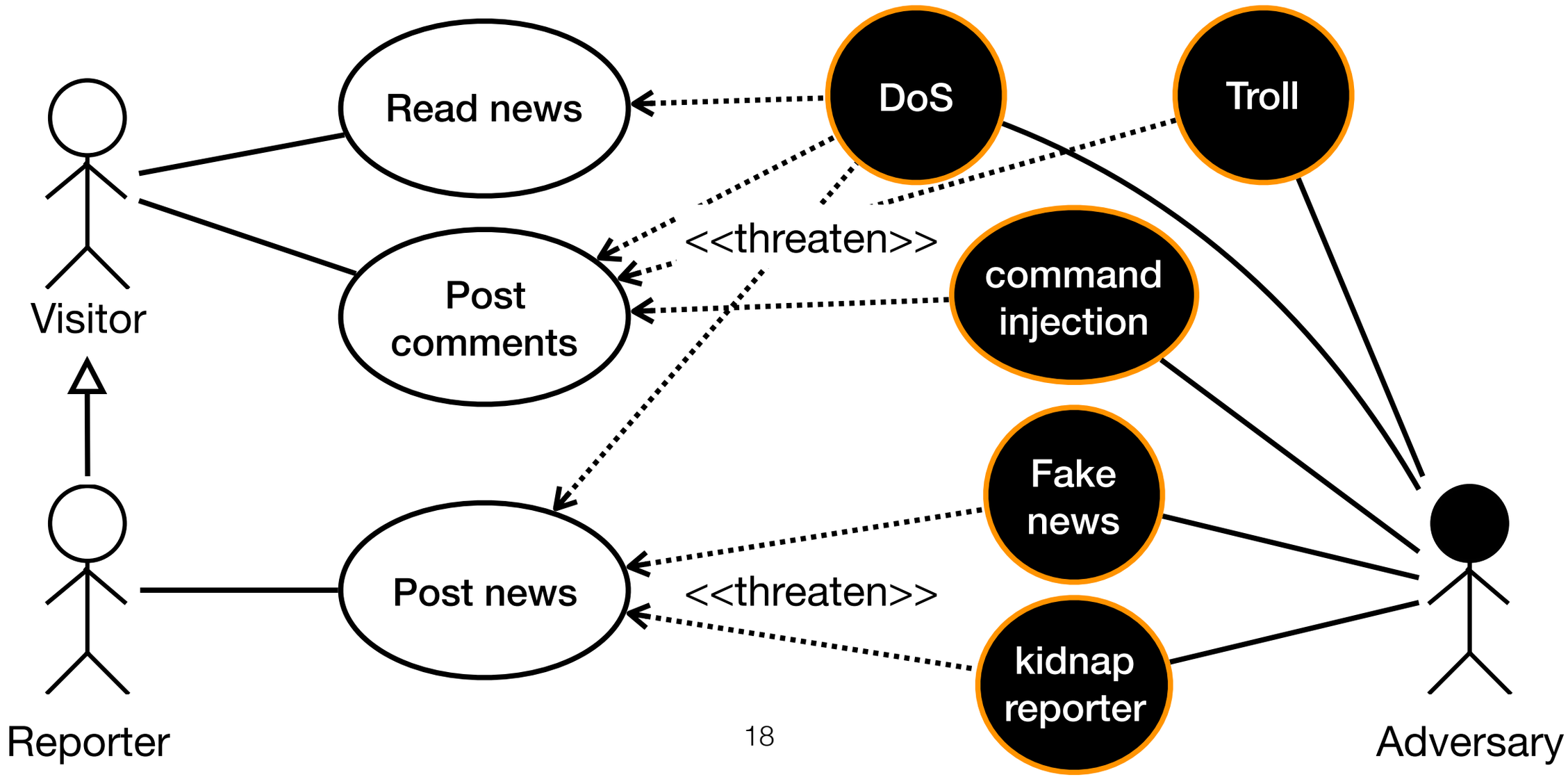
Example: News Website

- Security objectives: Integrity, availability of the service, safe place to visit for visitors

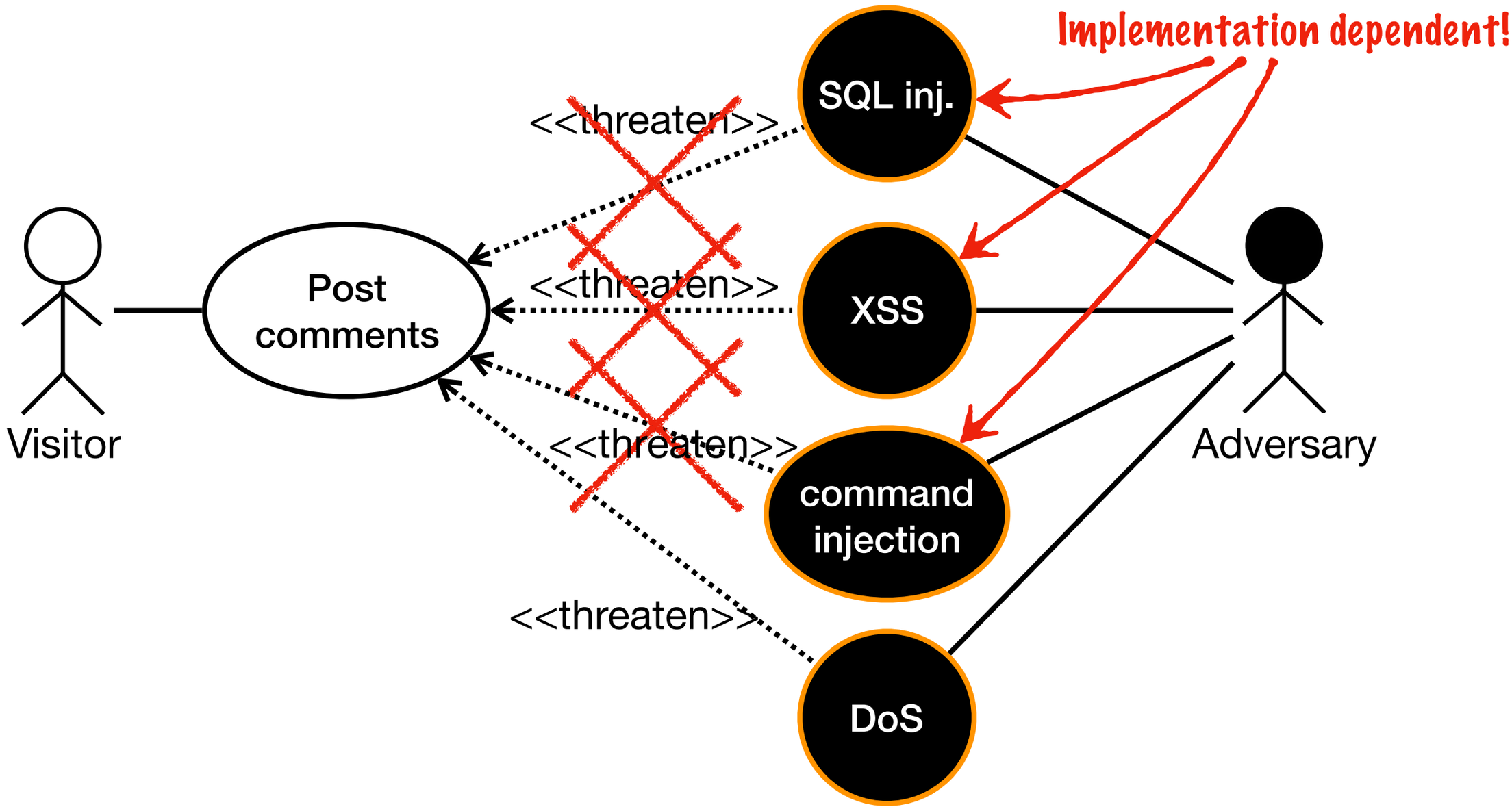


Example: News site

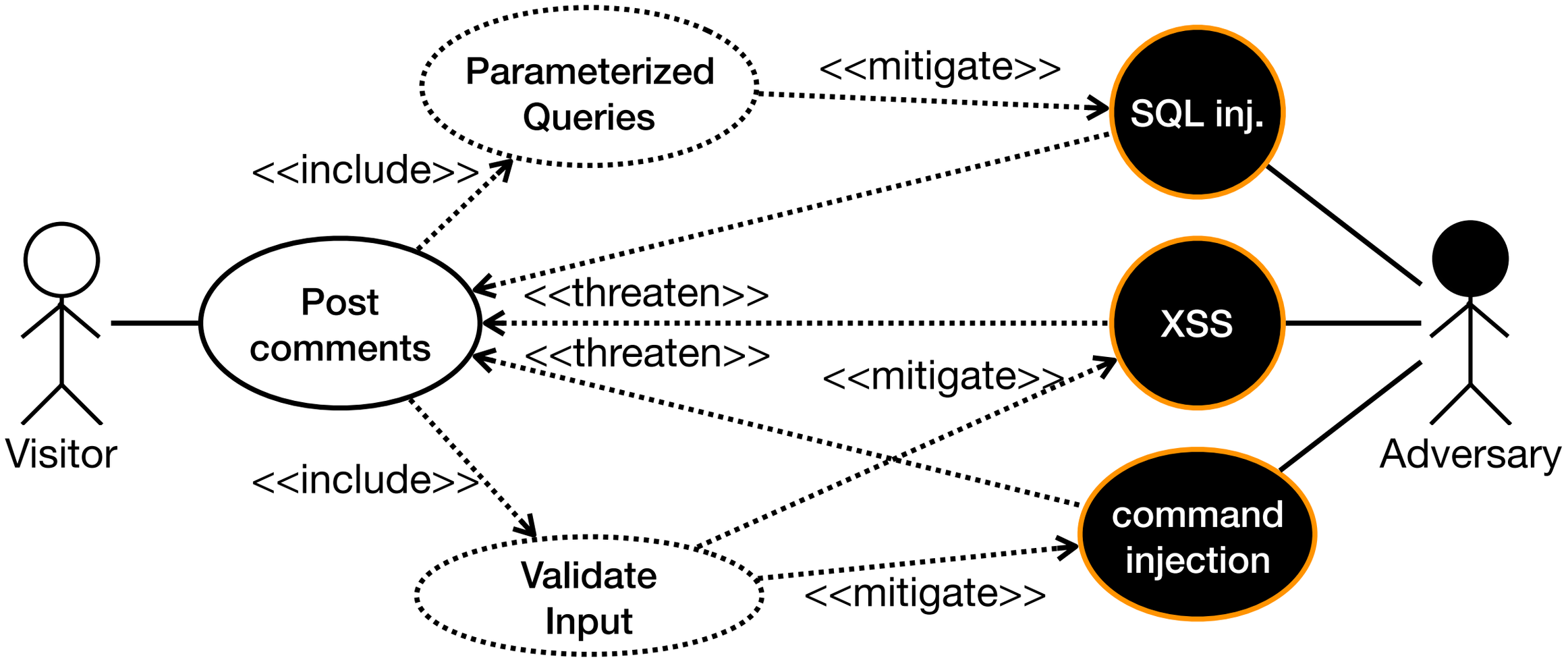
- Security objectives: Integrity, availability of the service, safe place to visit for visitors



Misuse of “Post comments”



Implementation-dependent threats

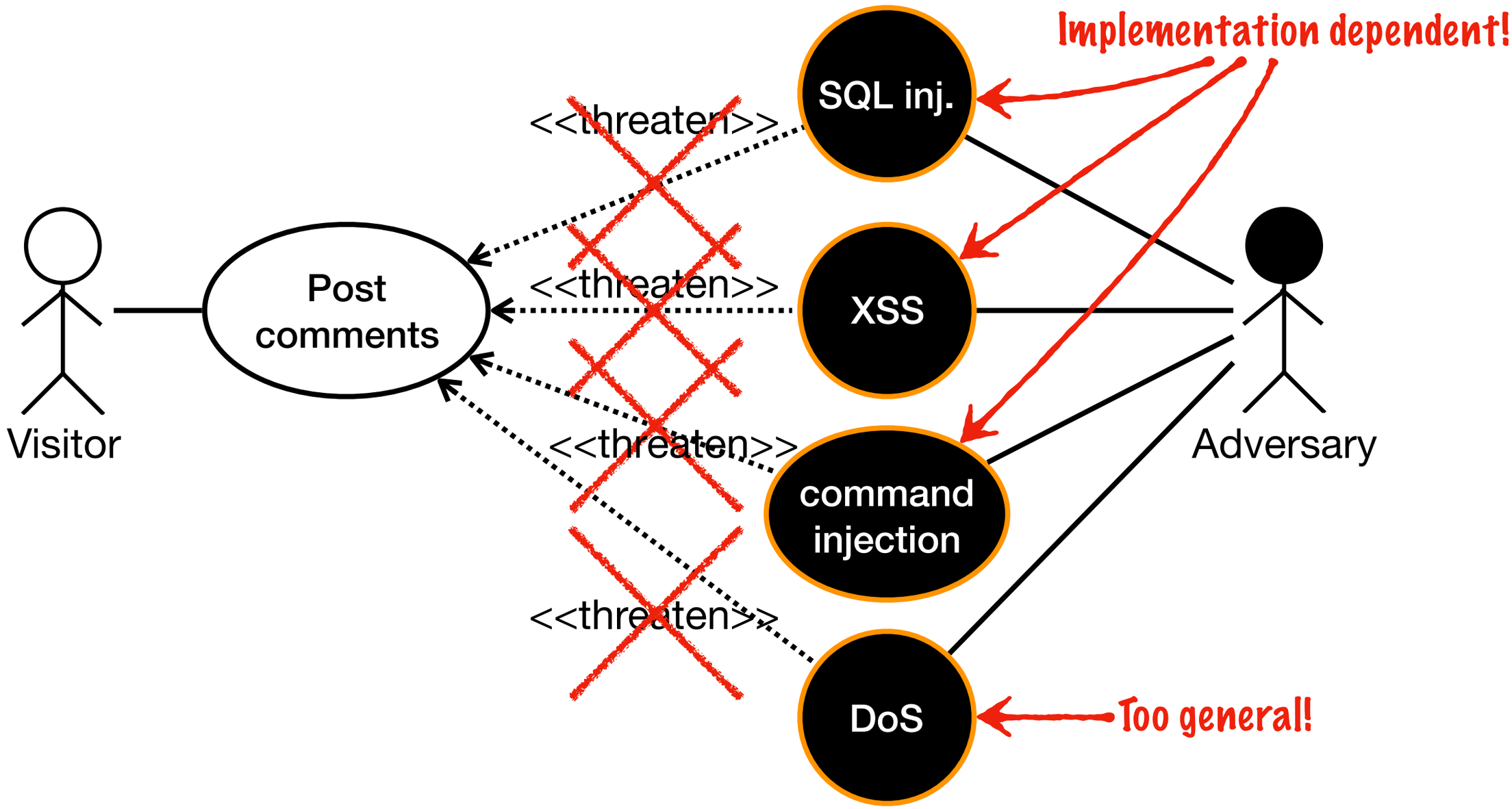


Important, but

- at wrong level of abstraction, do not contribute insight
- well-known, divert attention from architectural issues

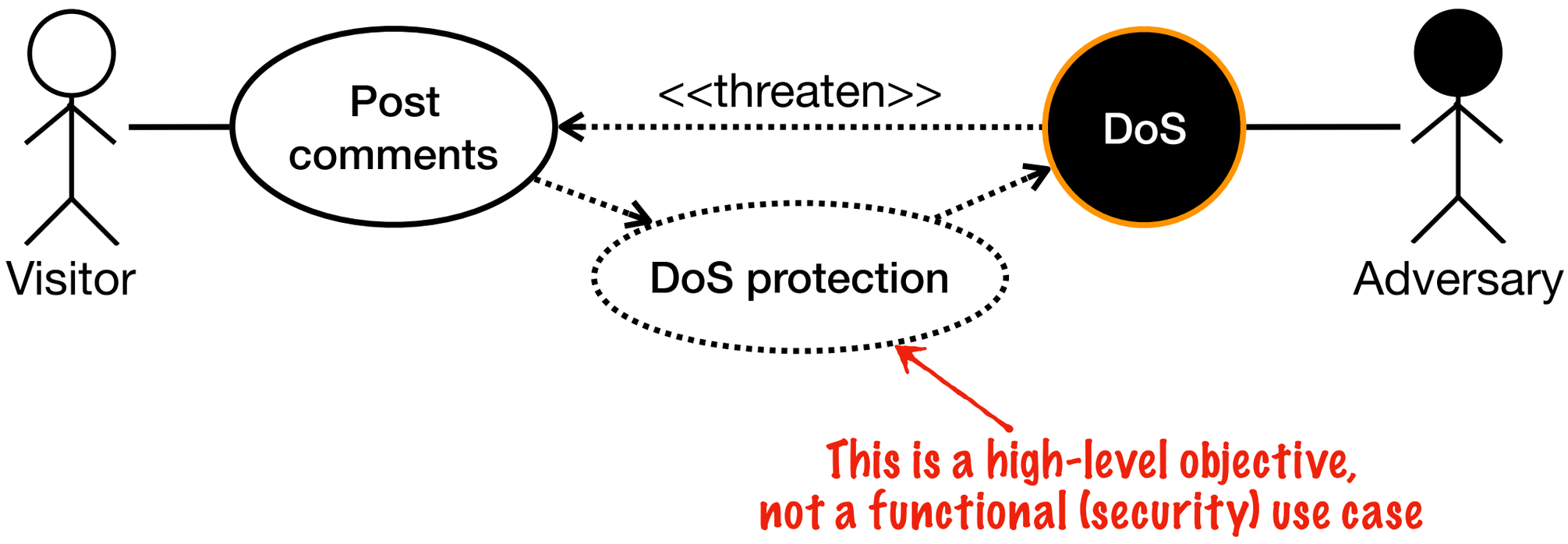
thus **ill-suited for the requirements elicitation stage**.

Misuse of “Post comments”



Overly general threats

Analysis process has terminated prematurely.

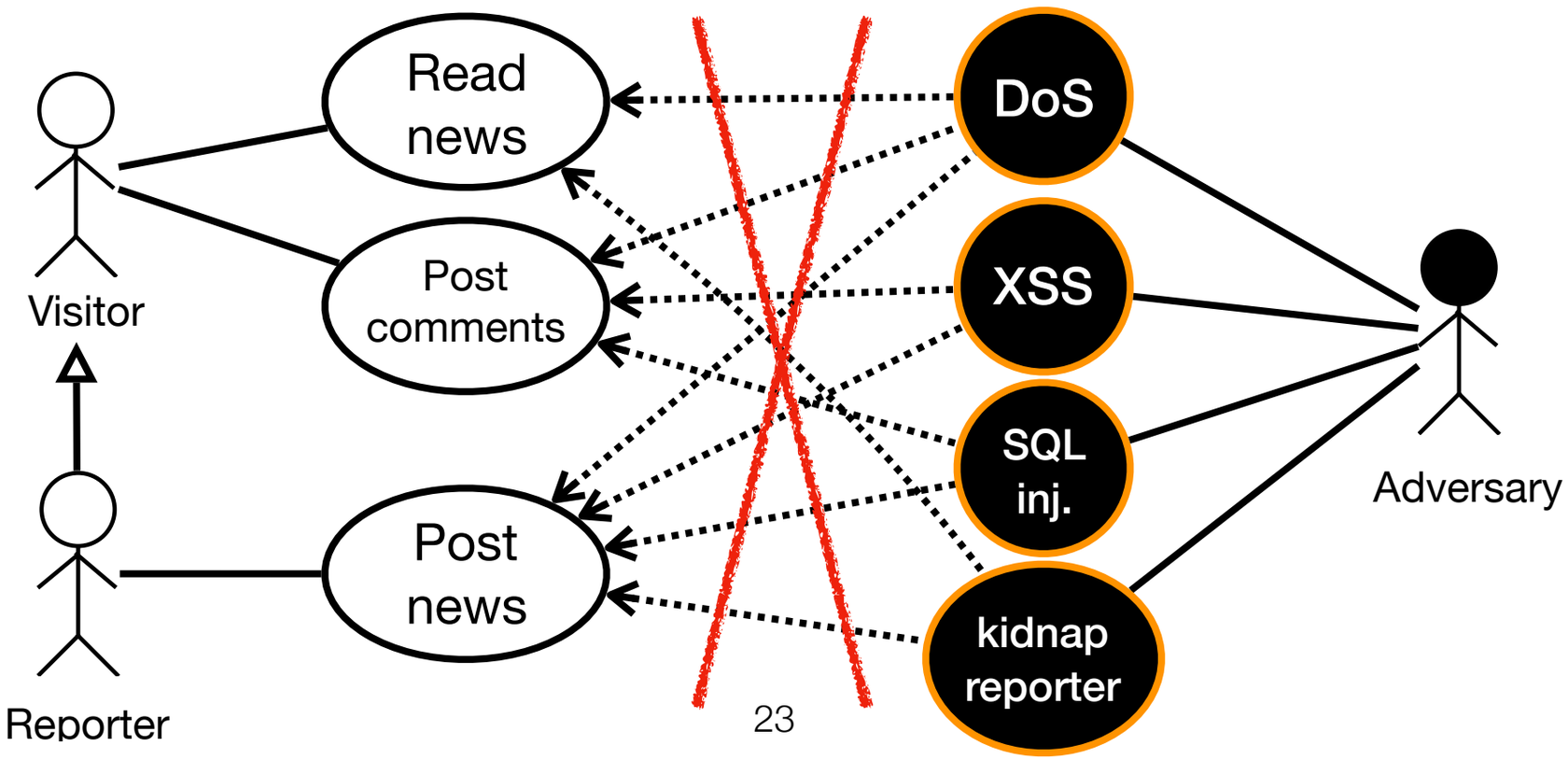


Continue the analysis!

Perhaps excessive posting of comments is an issue?

Negative Consequences of Orphan Misuse Cases

- **Use cases** elicited for the system are **ignored**.
- **Trade-offs** between security and cost, efficiency, usability, … remain **hidden**.
- **No insight** into system itself is provided, because
  - analysis ends prematurely at objectives level
  - well-known code-level attacks are listed



Avoiding Orphan Misuse Cases

- **Include:**
  - How can the use cases, as given, be misused?
  - Revise misuse cases that do not include any or categorically “threaten” (almost) all use cases.

- **Go bottom-up:** Start with a use case, not with an adversary.

- **Respect the abstraction level:**
  - Ignore implementation-dependent issues, skip well-known vulnerabilities.
  - Refine high-level misuse cases.

- **Make risks observable:** If no adversarial capability, value of protected assets, and cost-benefit questions appear, you are likely ignoring the system at hand.

Misuse of “Post comments”

Possible Mitigations:

1. Users must authenticate to post comments.
2. Comments are moderated by reporters before they become public.

Which mitigation to choose?

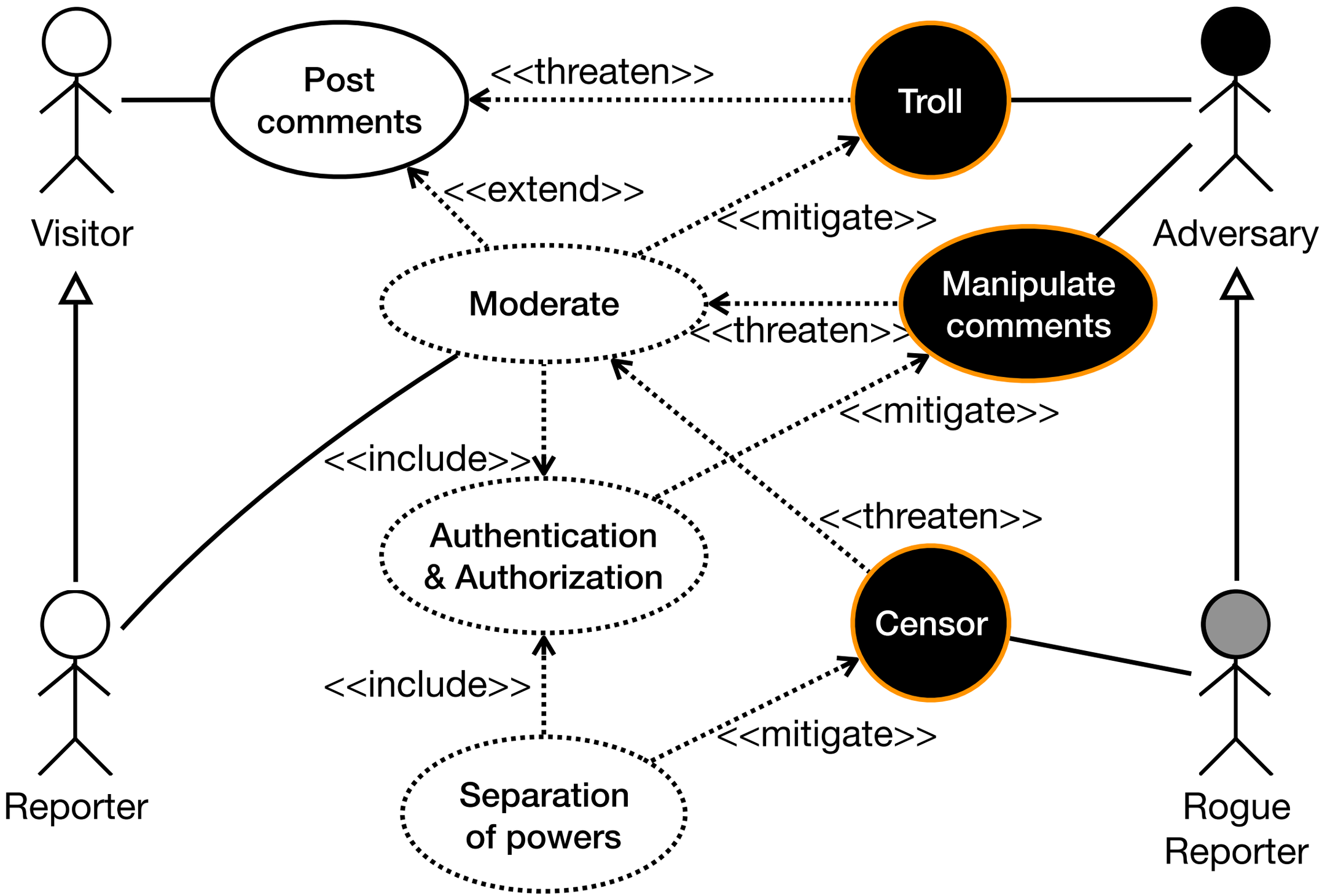
Mitigation 1: Drawback: Trolls will create new accounts once they are banned.

Mitigation 2: Drawback: Reporters could abuse moderation for censorship. This misuse needs a separation of powers mitigation.

- Question cannot be answered by requirements engineers alone.
- Communication with stake-holders necessary, options and tradeoffs need to be explained.

This misuse case is **focused** on the application, raises questions about adversarial capabilities, forces cost-benefit analysis, thus contributes to the creation of a more robust **architecture**.

Misuse Cases



Avoiding orphan misuse cases is not trivial

- Eliciting misuse cases that are specific to the system is harder than orphan misuse cases,
- literature gets it “wrong” (see paper for examples),
- we get it wrong (may find examples in this presentation),
- experience matters.

However:

- (Empirical evidence): Quality of misuse cases by our students improves, analysis paralysis lessens, after exposure to the anti-pattern.

Conclusion

Avoid orphan misuse cases!

Some threats will be disregarded.

This is a desirable limitation:

- Impossible to cover all security issues. We must avoid analysis paralysis. Avoiding orphan misuse cases helps to focus on architectural issues.

- There is little we can do about implementation issues in the requirements elicitation phase. Code review and testing phase is responsible for this.

Future Work

- Evaluate our approach and misuse case analysis method.
- Identify root cause of each flaw in a system. What are the deficiencies in the requirements engineering and other development phases?
- Requires substantial effort for an industrial-scale analysis.