BASICS OF SECURITY REQUIREMENT ELICITATION

Shahriyar Jalayeri

What is Requirment Engineering?

1. What a system must do
2. Known limitation and constrains on the resouse or design
3. How well system must do (1)

- Functional Requerment : (1)
  - Specific, Unambiguous, Measurable, Obseravle, Testable
- Non-functional Requirment : (2) and (3)
  - *E.g : Safety, Security, Availibily, Reliability, Cost, Quality, …*

What are the steps in RE?

- Elicitation
  - *Interview, Brainstroming, Role Playing, Prototyping, Games, …*
- Analysis
  - Consistent and fit together, not contradicting each other
- Specification and Documentation
  - *NL, Graphical (e.g. UML), Mathematical*
- Validation and Verification

Specification

- UML, Pseudocode, Math
  - Need of special training
  - Translation Error
- *Natural Language*
  - Unconstrained use,inherently unsuitable for requirements definition
    - Ambiguity (a word or phrase has two or more different meanings)
    - Vagueness (lack of precision, structure and/or detail)
    - Complexity
    - Untestability (cannot be proven true or false when the system is implemented)

Introducing EARS

- Easy Approach to Requirment Syntax (EARS)
- Two cases of requirment
  - *Normal Behavior*
    - When everything works fine
  - *Unwanted Behavior*
    - When there is error, failure, malfunctioning

Generic Requirements Syntax

- *Template*

  <optional pre-condition> <optional trigger> the <system name> shall <system response>

- *Pre-condition : conditions in which the requirement can be invoked*
- *Trigger : event that initiates the requirement*
- *System response: the necessary system behaviour*

Generic Requirements Syntax (cont.)

- *Template*

  <optional pre-condition> <optional trigger> the <system name> shall <system response>

- *Order is significatnt, follows temporal logic:*
  - *Any preconditions must be satisfied otherwise the requirement cannot ever be activated.*
  - *The trigger must be true for the requirement to be “fired”, but only if the preconditions were already satisfied.*
  - *The system is required to achieve the stated system response if and only if the preconditions and trigger are true.*

Normal Behavior T1 : Ubiquitous

- Defines the system behavior that is active all the time, it is “continious”.
- No pre-condition or trigger, it is “unconditional”.
- Examples
  - *The car shall have maximum retail price of XXX.*
  - *The laptop shall have a maximum mass of XXX grams.*
  - *The laptop shall have minimum XXX hours of battery life.*
  - *The monitor shall have minimum XXX lumens of brightness.*

Normal Behaviour T2 : Event-driven

- Syntax

  When <trigger> the <system name> shall <system response>

- Req is initiated only when a triggering event is detected within the system boundary.
- The trigger is something that the system itself can detect.
- Examples
  - *When a process runs out of memory the OS shall kill the process.*
  - *When a packet with ACK message is recived the OS shall respond with a SYN message.*
  - *When the laptop is turned off and the power botton in pressed the laptop shall boot up.*
  - *When the process is in idle state and the process recives a signal the process shall log the signal number ~~to a file under /etc/signal.log~~ .*

Why?

Normal Behavior T3 : State-driven

- Syntax

  While <in specific state> the <system name> shall <system response>

- *Req is activated when the system is in a defined state, req is “cont” but only while the system is in that specific defined state.*
- *Examples*
  - *While the ignition is on, the cas shall display the fule level and oil level to the driver.*
  - *While thekey is in the car, the car alarm shall be disabled.*
  - *While the laptop is running on battery and battery is less than 10 precent, the laptop shall display “low battery” message..*

Normal Behavior T4: Option

- Syntax

  Where <feature is included> the <system name> shall <system response>

- Req is applicable only when a system includes the particular feature.
- Examples
  - Where the car has electric windows, the car windows control button shall be on the driver door panel.

Unwanted Behavior

- Syntax

  if <optional per-condition> <trigger>, then the <system name> shall <system response>

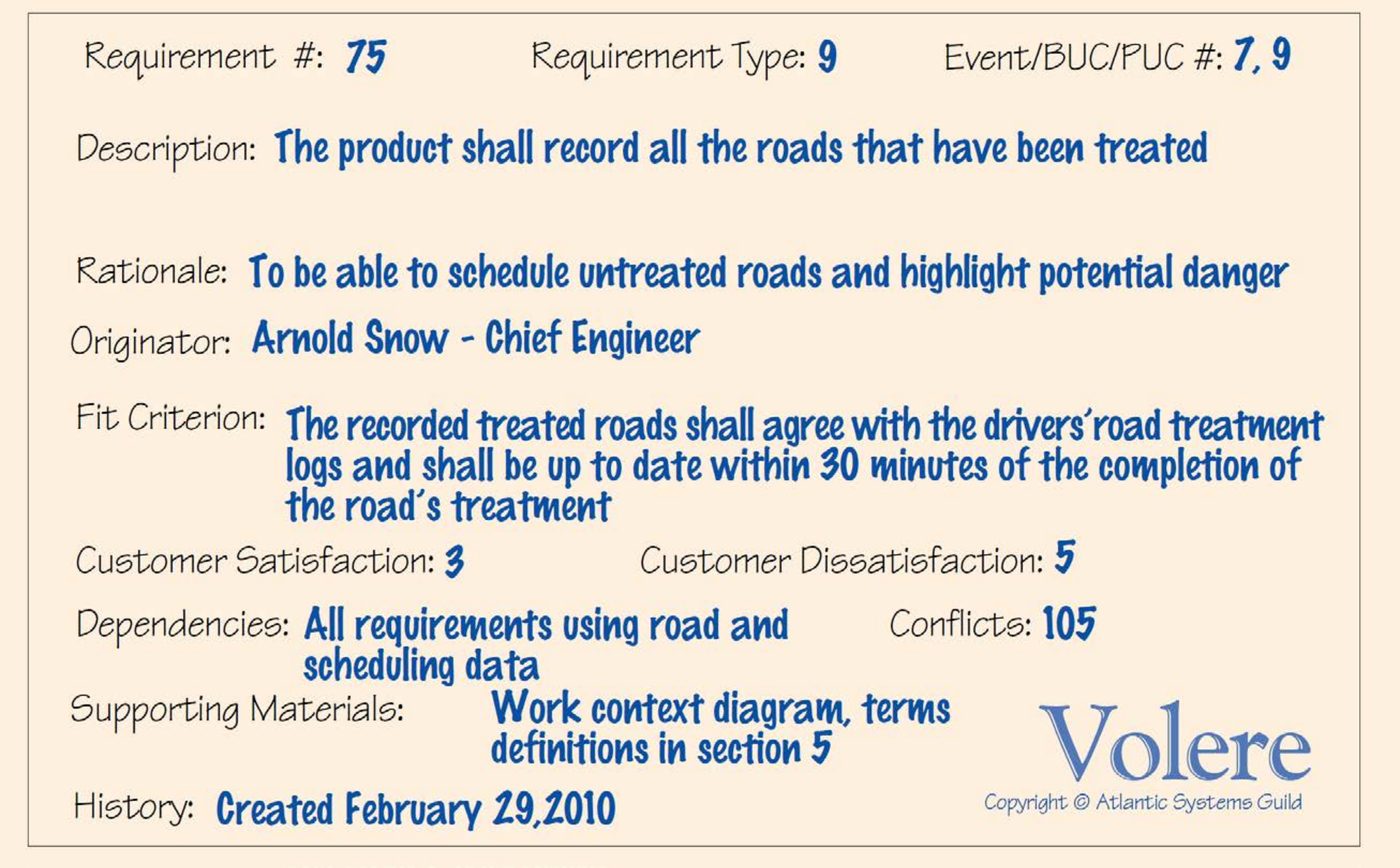
- Major source of omission
- Variant of event driven requirement
- Given their own syntax, to be easily identified throughout the lifecycle
- Examples
  - *If the car detects attempted intrusion, then the car shall activate the car alarm.*
  - *If tampering with the RO root file-system is detects, then the system shall not boot.*
  - *If incorrect password entered more than 5 times, then the laptop shall wait 5 seconds before asking for password again.*
  - *If the device is flashed with lower software version, then the device shall show a warning that system is running with lower version and stop the boot process.*

System response mitigates the impact of the unwanted event, or prevents the system from entering an unwanted state.

Complex Requirement

- Requirement with complex conditional clauses,
- defined using combination of When, Where, While, If-Then
- Example
  - *While the laptop is operating on main electrical power, if the power cable is disconnected, then the laptop shall display and warning message.*

Volere Requirements Specification Template



Requirement #: 75 Requirement Type: 9 Event/BUC/PUC #: 7, 9

Description: The product shall record all the roads that have been treated

Rationale: To be able to schedule untreated roads and highlight potential danger

Originator: Arnold Snow - Chief Engineer

Fit Criterion: The recorded treated roads shall agree with the drivers'road treatment logs and shall be up to date within 30 minutes of the completion of the road's treatment

Customer Satisfaction: 3 Customer Dissatisfaction: 5

Dependencies: All requirements using road and scheduling data

Conflicts: 105

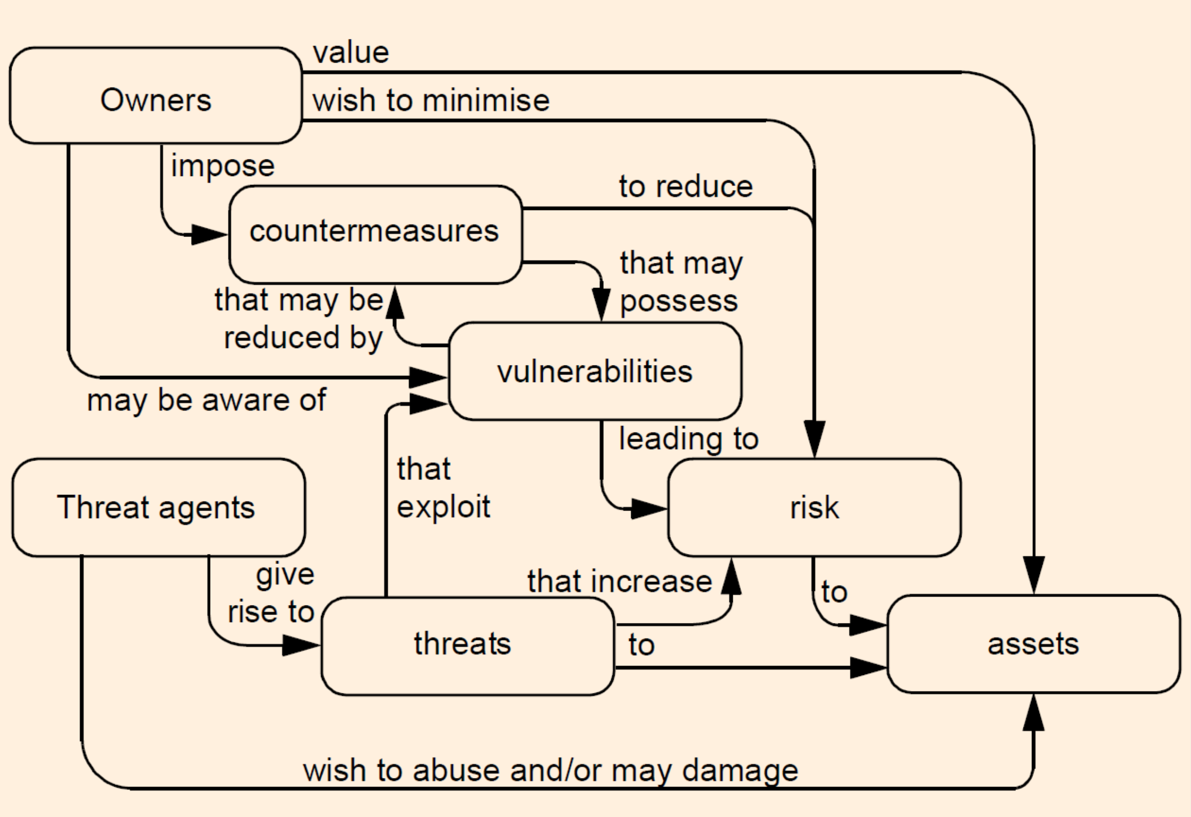
Supporting Materials: Work context diagram, terms definitions in section 5

History: Created February 29,2010

Volere

Eliciting Security Requirements

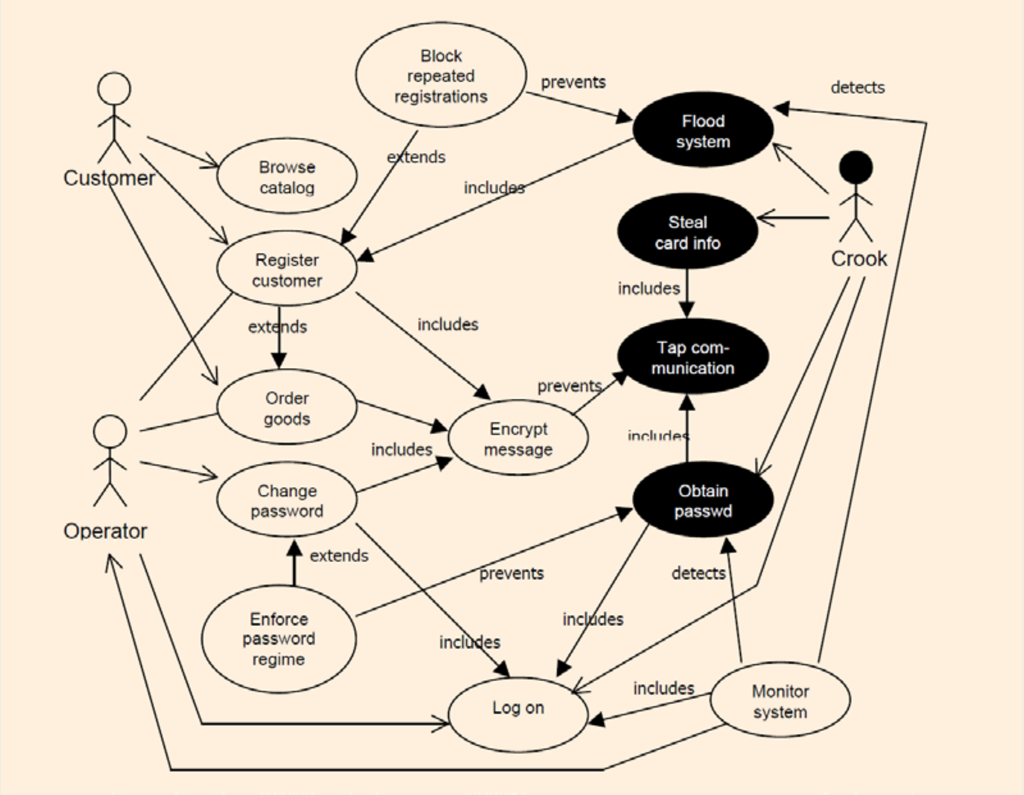
- Start by understanding the problem (not solution)



Abuse/Misuse Cases

- *Abuse Case (McDermott & Fox, 1999)*
- *Misuse cases (Sindre & Opdahl, 2001)*
- *Threat Modeling (Myagmar Et al. , 2005)*

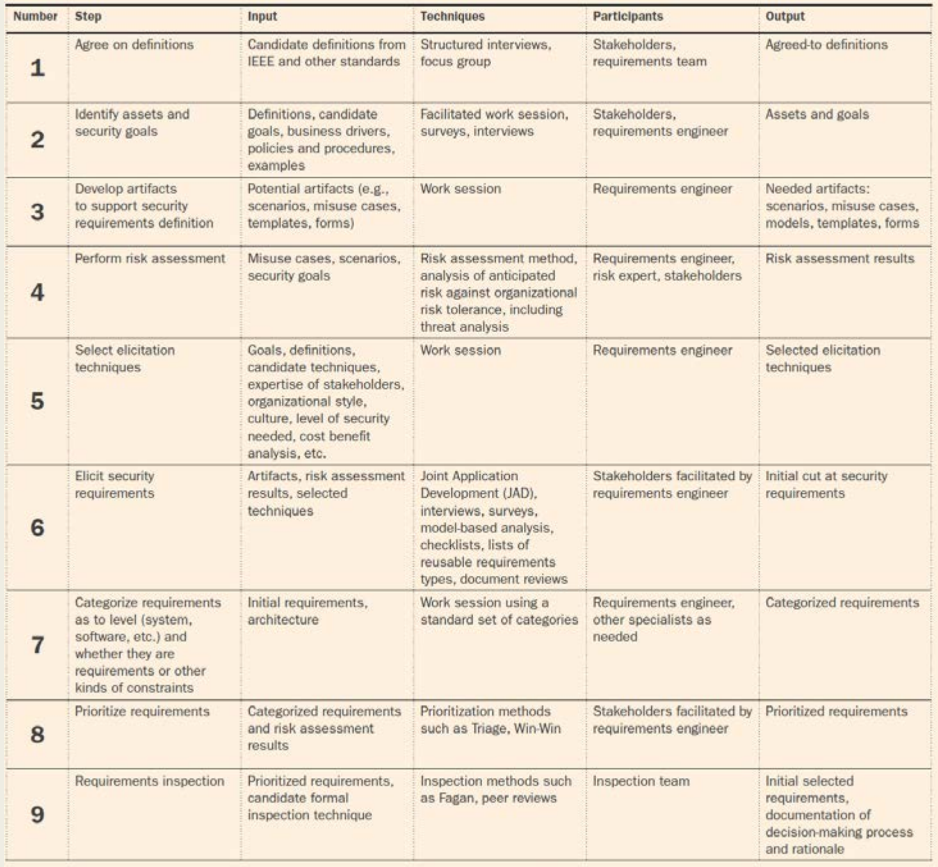
- The Security Requirement Process
  - *Identify critical asset*
  - *Define security goal (e.g. Confidentiality)*
  - *Identify threats*
  - *Identify and analyze risk*
  - *Define security requirement*



Misuse case, Misuser and threaten use case

SQUARE Methodology

- Developed by Mead & Stehney, 2005
- Consist of a 9-step process:
  - *Step 1: Agree on definitions*
  - *Step 2: Identify security goals*
  - *Step 3: Develop supporting artifacts*
  - *Step 4: Perform risk assessment*
  - *Step 5: Select elicitation techniques*
  - *Step 6: Elicit security requirements*
  - *Step 7: Categorize requirements*
  - *Step 8: Prioritize requirements*
  - *Step 9: Requirements inspections*

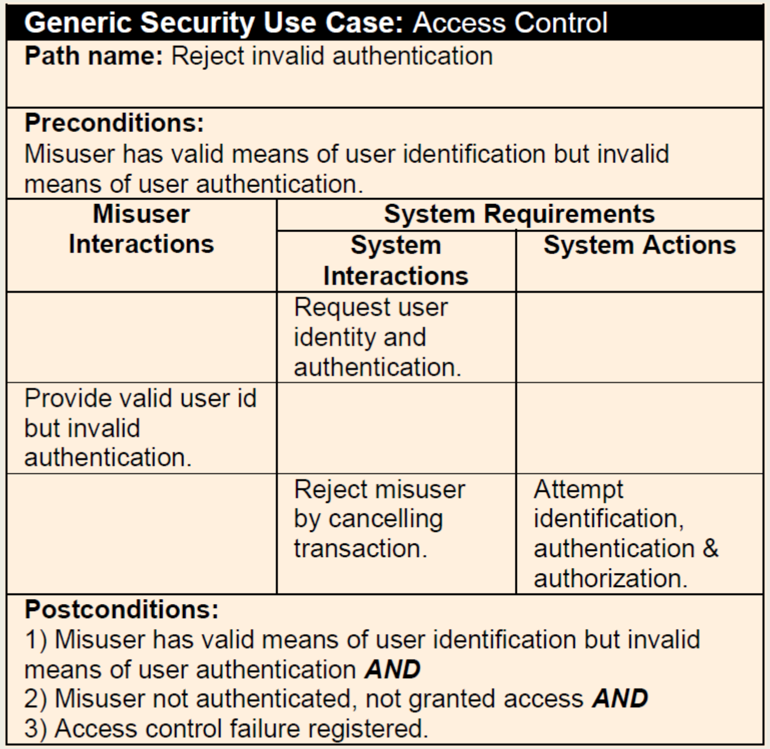


Number	Step	Input	Techniques	Participants	Output
1	Agree on definitions	Candidate definitions from IEEE and other standards	Structured interviews, focus group	Stakeholders, requirements team	Agreed-to definitions
2	Identify assets and security goals	Definitions, candidate goals, business drivers, policies and procedures, examples	Facilitated work session, surveys, interviews	Stakeholders, requirements engineer	Assets and goals
3	Develop artifacts to support security requirements definition	Potential artifacts (e.g., scenarios, misuse cases, templates, forms)	Work session	Requirements engineer	Needed artifacts: scenarios, misuse cases, models, templates, forms
4	Perform risk assessment	Misuse cases, scenarios, security goals	Risk assessment method, analysis of anticipated risk against organizational risk tolerance, including threat analysis	Requirements engineer, risk expert, stakeholders	Risk assessment results
5	Select elicitation techniques	Goals, definitions, candidate techniques, expertise of stakeholders, organizational style, culture, level of security needed, cost benefit analysis, etc.	Work session	Requirements engineer	Selected elicitation techniques
6	Elicit security requirements	Artifacts, risk assessment results, selected techniques	Joint Application Development (JAD), interviews, surveys, model-based analysis, checklists, lists of reusable requirements types, document reviews	Stakeholders facilitated by requirements engineer	Initial cut at security requirements
7	Categorize requirements as to level (system, software, etc.) and whether they are requirements or other kinds of constraints	Initial requirements, architecture	Work session using a standard set of categories	Requirements engineer, other specialists as needed	Categorized requirements
8	Prioritize requirements	Categorized requirements and risk assessment results	Prioritization methods such as Triage, Win-Win	Stakeholders facilitated by requirements engineer	Prioritized requirements
9	Requirements inspection	Prioritized requirements, candidate formal inspection technique	Inspection methods such as Fagan, peer reviews	Inspection team	Initial selected requirements, documentation of decision-making process and rationale

Security Requirements Elicitation and Analysis Process

Reusable Security Requirements

- Define generic assets, threats, tests and mitigation
- Construct a security requirement repository
  - STRIDE and CAPEC



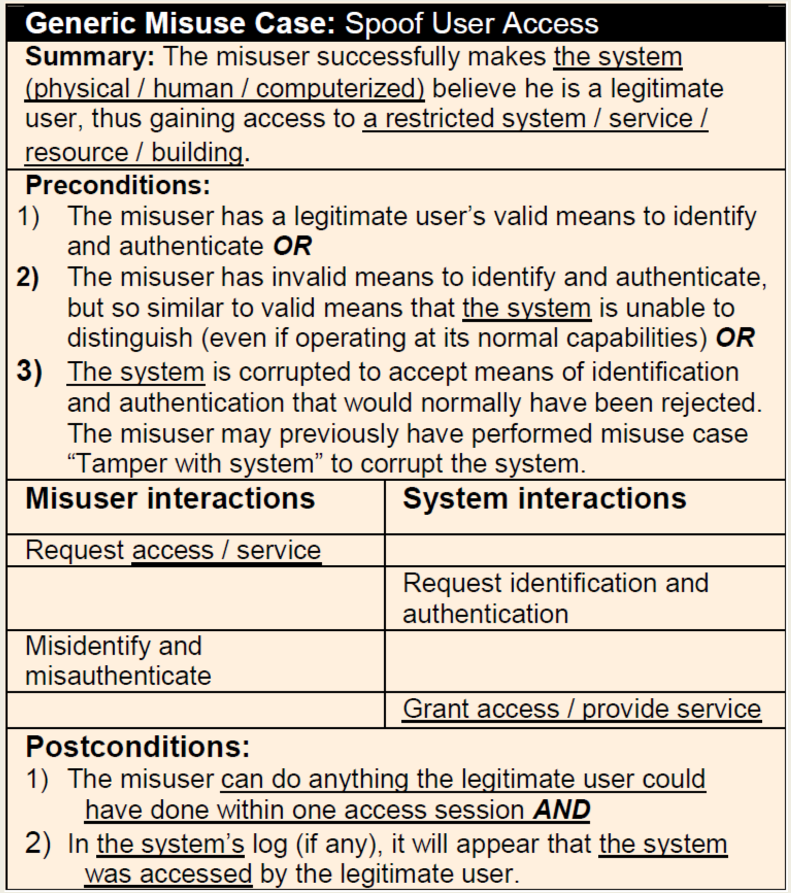
Generic Security Use Case: Access Control

Path name: Reject invalid authentication

Preconditions:
Misuser has valid means of user identification but invalid means of user authentication.

Misuser Interactions	System Requirements	
	System Interactions	System Actions
	Request user identity and authentication.	
Provide valid user id but invalid authentication.		
	Reject misuser by cancelling transaction.	Attempt identification, authentication & authorization.

Postconditions:
1) Misuser has valid means of user identification but invalid means of user authentication ***AND***
2) Misuser not authenticated, not granted access ***AND***
3) Access control failure registered.



Generic Misuse Case: Spoof User Access

Summary: The misuser successfully makes the system (physical / human / computerized) believe he is a legitimate user, thus gaining access to a restricted system / service / resource / building.

Preconditions:
1) The misuser has a legitimate user's valid means to identify and authenticate ***OR***
2) The misuser has invalid means to identify and authenticate, but so similar to valid means that the system is unable to distinguish (even if operating at its normal capabilities) ***OR***
3) The system is corrupted to accept means of identification and authentication that would normally have been rejected. The misuser may previously have performed misuse case "Tamper with system" to corrupt the system.

Misuser interactions	System interactions
Request access / service	
	Request identification and authentication
Misidentify and misauthenticate	
	Grant access / provide service

Postconditions:
1) The misuser can do anything the legitimate user could have done within one access session ***AND***
2) In the system's log (if any), it will appear that the system was accessed by the legitimate user.

References

- *Mavin, Alistair, et al. "Easy approach to requirements syntax (EARS)." 2009 17th IEEE International Requirements Engineering Conference. IEEE, 2009.*
- *Kausar, Sumaira, et al. "Guidelines for the selection of elicitation techniques." 2010 6th International Conference on Emerging Technologies (ICET). IEEE, 2010.*
- *Gunda, Sai Ganesh. "Requirements engineering: elicitation techniques." (2008).*
- *Sindre, Guttorm, and Andreas L. Opdahl. "Capturing security requirements through misuse cases." NIK 2001, Norsk Informatikkonferanse 2001, http://www. nik. no/2001 74 (2001).*
- *Sindre, Guttorm, and Andreas L. Opdahl. "Eliciting security requirements with misuse cases." Requirements engineering 10.1 (2005): 34-44.*
- *McDermott, John, and Chris Fox. "Using abuse case models for security requirements analysis." Proceedings 15th Annual Computer Security Applications Conference (ACSAC'99). IEEE, 1999.*
- *Firesmith, Donald. "Engineering security requirements." J. Object Technol. 2.1 (2003): 53-68.*
- *Firesmith, Donald. "Specifying reusable security requirements." J. Object Technol. 3.1 (2004): 61-75.*
- *Myagmar, Suvda, Adam J. Lee, and William Yurcik. "Threat modeling as a basis for security requirements." Symposium on requirements engineering for information security (SREIS). Vol. 2005. 2005.*
- *Peeters, Johan. "Agile security requirements engineering." Symposium on Requirements Engineering for Information Security. Vol. 12. 2005.*
- *Toval, Ambrosio, et al. "Requirements reuse for improving information systems security: a practitioner’s approach." Requirements Engineering 6.4 (2002): 205-219.*
- *Sindre, Guttorm, Donald G. Firesmith, and Andreas L. Opdahl. "A reuse-based approach to determining security requirements." REFSQ. Vol. 3. 2003.*