



# Secure Software Development

## Abuse Cases

### Objectives

*Already covered*  
*Abusing SW*

- Differentiate between security goals and security functions
- Explain the concept of a security requirement
- Compare and contrast requirements and anti-requirements
- Explain the concept of an abuse case and explain how one would create an abuse case
- Explain what you need to do to think securely.
- Explain the concept of a use case diagram
- Explain the concept of abuse cases as shown on a use case diagram
- Construct examples of abuse cases

# Example confidentiality requirements

- Personal health information must be protected against disclosure using approved encryption mechanisms
- Passwords and other sensitive input fields need to be masked
- The use of nonsecure transport protocols such as File Transfer Protocol (FTP) to transmit account credentials in clear to third parties outside of your organization shall not be allowed.

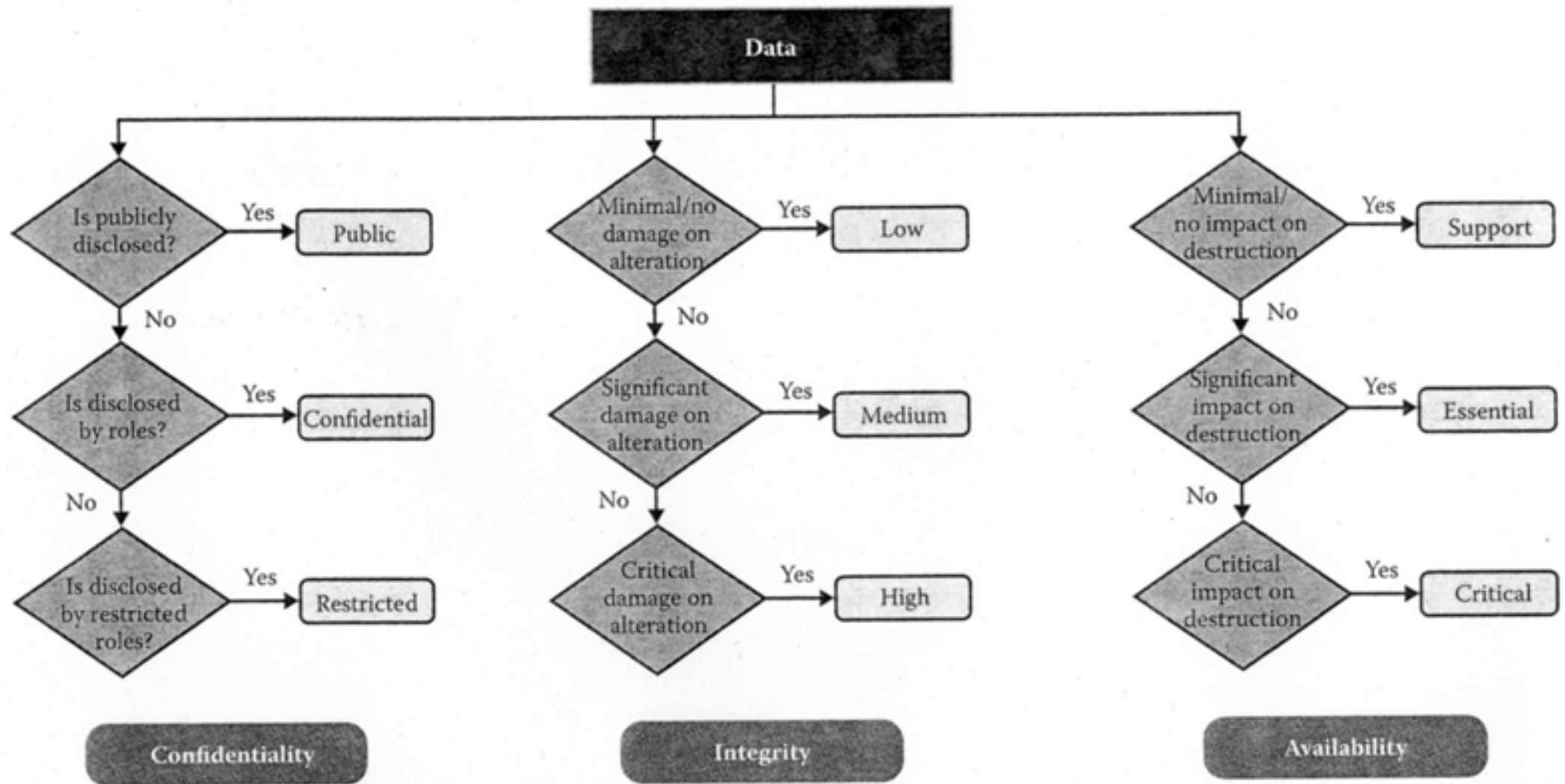
# Integrity Requirements

- Address reliability assurance and protection against unwanted modification
  - Needs to deal with both system and data integrity
- May use one or more of the following
  - Input validation
  - Parity bit checking
  - hashing

# Example Requirements

- All input forms and query strings shall be validated against a set of allowable inputs before the software accepts it for processing
- All messages transmitted over the internet shall be checked for corruption before being processed

# Categoryization of data



# Authentication requirements

- The process of validating an entity's claim
  - Are you who you say you are
- Two factor identification
  - Uses two factors for identification
  - Something someone knows
  - Something someone has

1.



Sign in with your  
**Google Account**

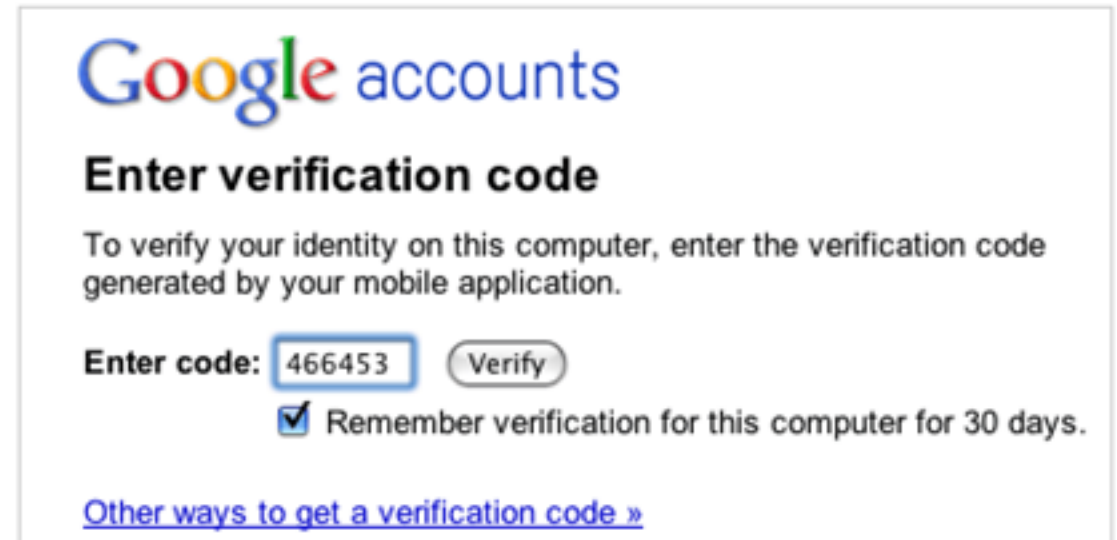
Email:   
ex: pat@example.com

Password:

Stay signed in

[Can't access your account?](#)

2.



**Google accounts**

**Enter verification code**

To verify your identity on this computer, enter the verification code generated by your mobile application.

Enter code:

Remember verification for this computer for 30 days.

[Other ways to get a verification code »](#)

- Multifactor authentication
  - Uses multiple factors for authentication

# Who are attackers?

Malicious Intent

Not standard customer

⇒ Smart / Creative

⇒ Always out there

→ trying to undermine

the system.

# Critical Thinking For Security

- Think like a bad person with malicious intent
  - Exploit your mistakes
- Exploit creativity
  - Boundary conditions
  - Edges
  - Intersystem communication
- Example: Design assumes connections from the web server to the database server are always valid
  - Hacker: Try to make the web server send erroneous requests to the database server



# Example: Cookies

- Web site sets a cookies on a local machine. Website reads that back.

What might a hacker do?  $\Rightarrow$

Change  
the cookie to  
sum of things I see

---

Item#	27
Price	<del>\$29.99</del> \$10.01
CRC1	

checksum

Abuse Cases

deleted a iteration



# Abuse Cases

- A specification of a type of complete interaction between a system and one or more actors, where the results of the interaction are harmful to the system, one of the actors, or one of the stakeholders in the system.

# How to Create Misuse Cases

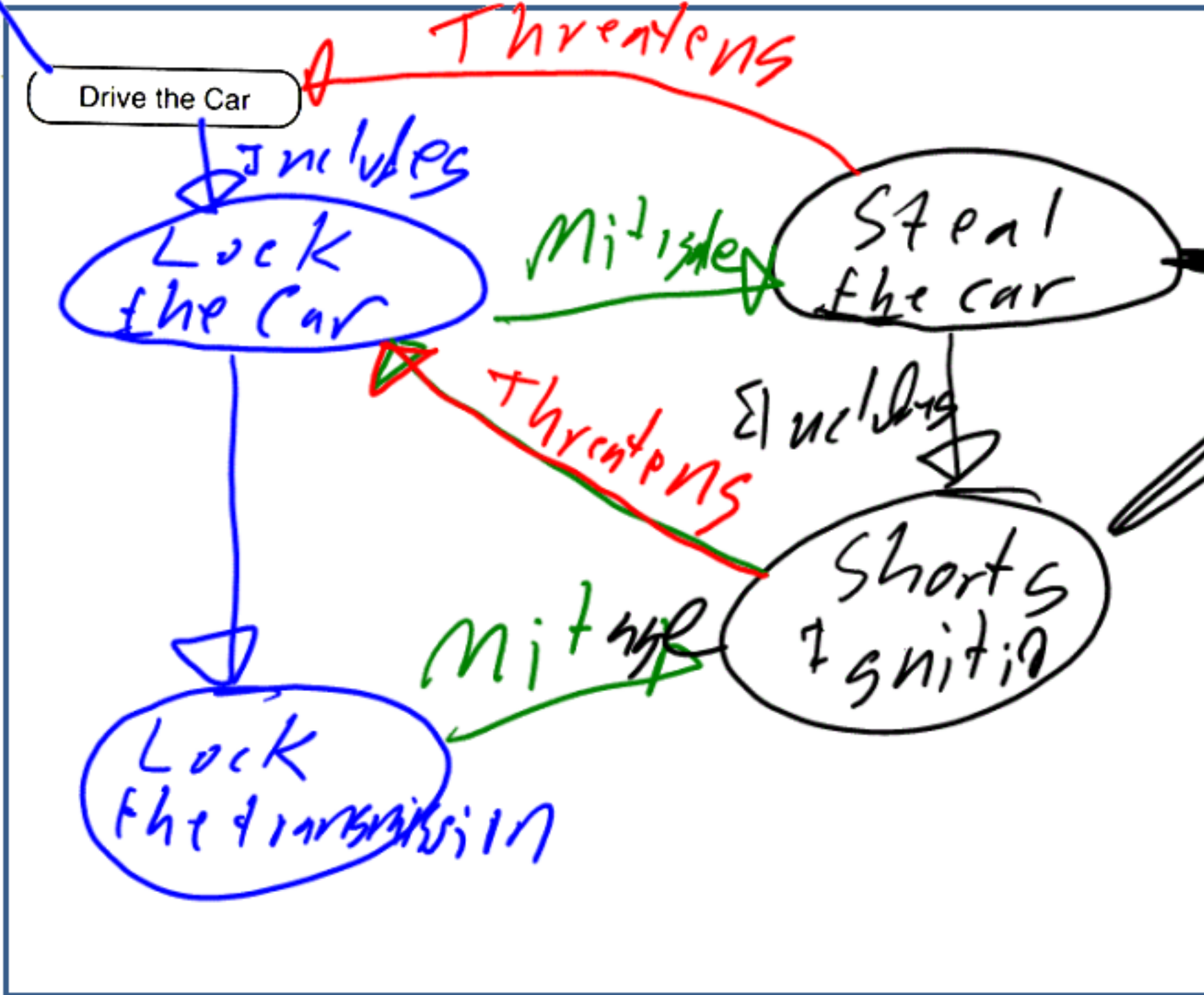
- A practical approach to create abuse cases is through informed brainstorming
- Don't believe "but no one would ever do that"
- Start by knowing the requirements, standard use cases, and attack patterns
- Identify and document threats (who may want to attack and why)
- Create anti-requirements
  - things that you don't want your software to do
- Create an attack model
  - cycle through attack patterns to see whether they apply

*Project Specific*



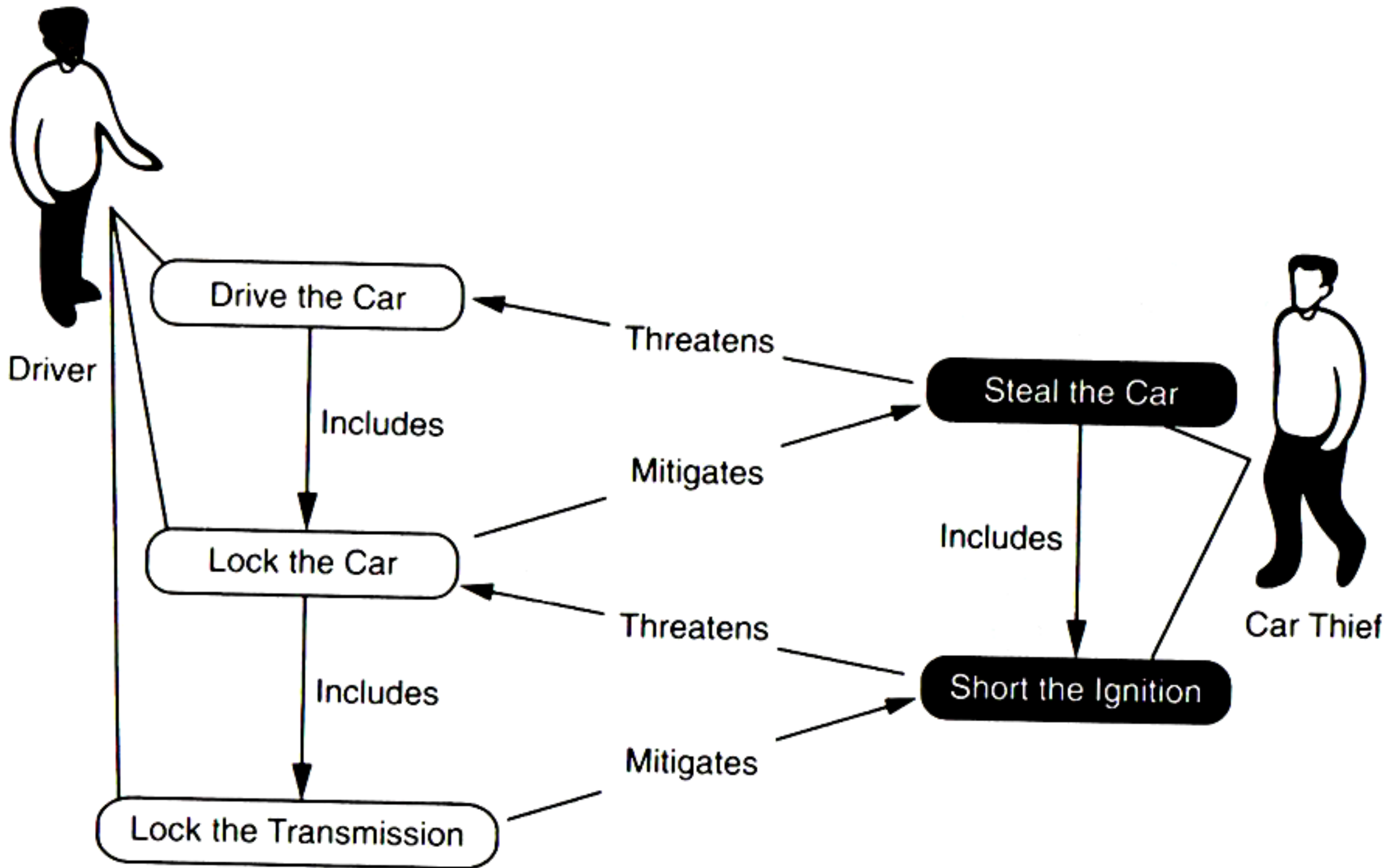
Driver

# Automobile

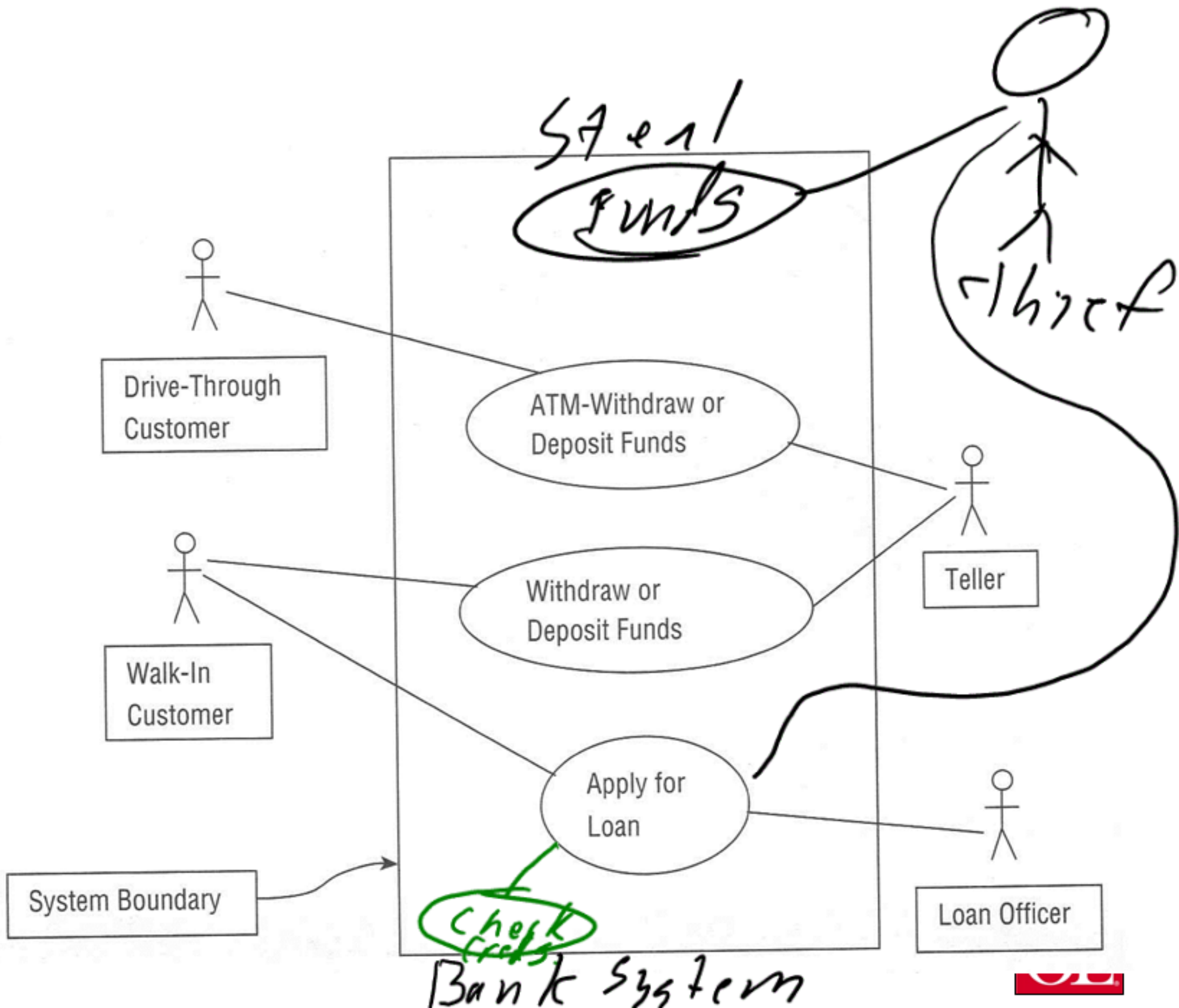


Thief





# Banking Use Case Diagram



# Anti-requirements

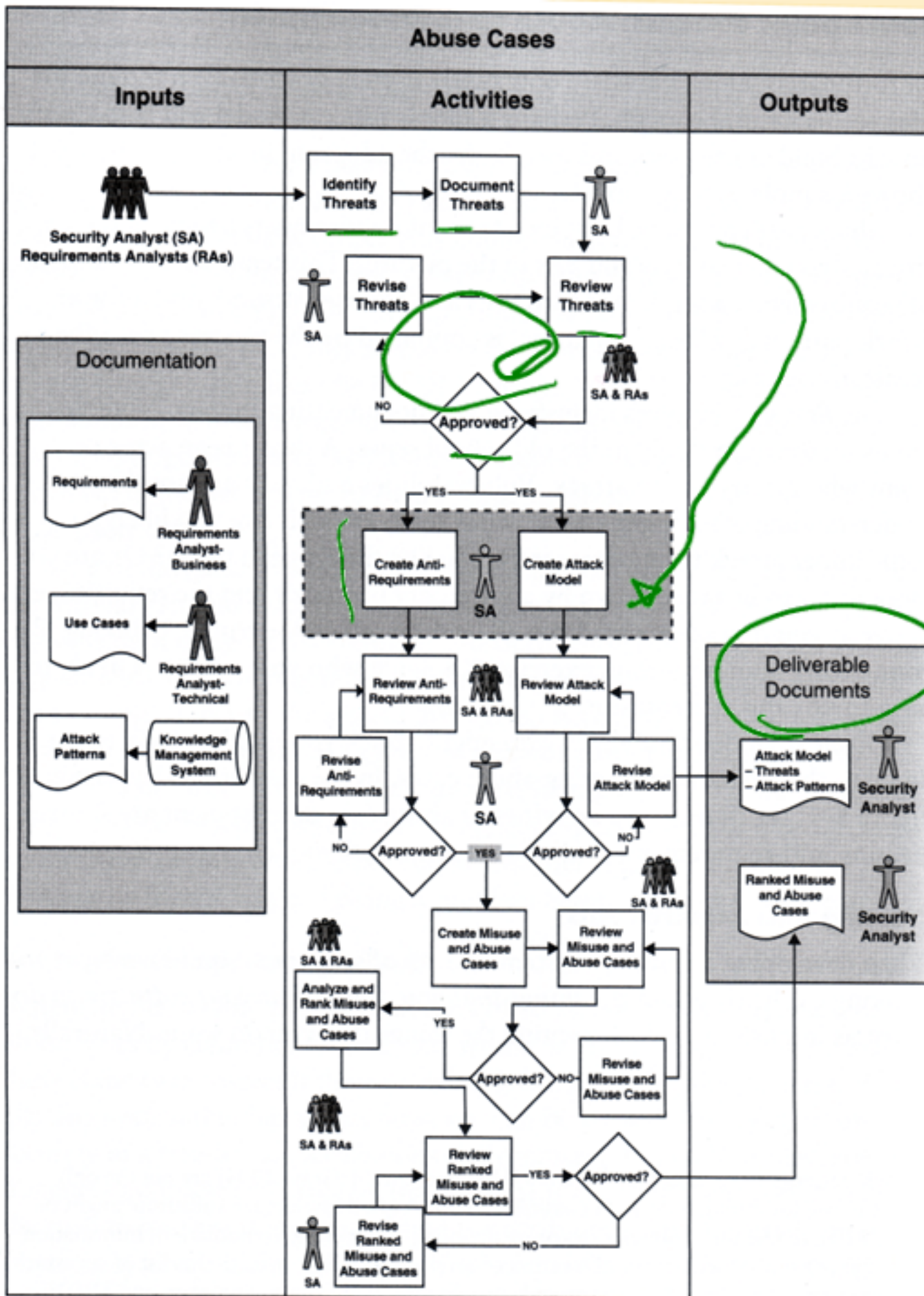
- Explicit things that you DO not want your software to do
- Generated by security analysis in conjunction with requirements analysts
- Generally described by the lack of a security function or failure of a security function



Abuse Cases

# Process for building abuse cases

## cases



*Deliverable Documents that can be reviewed*



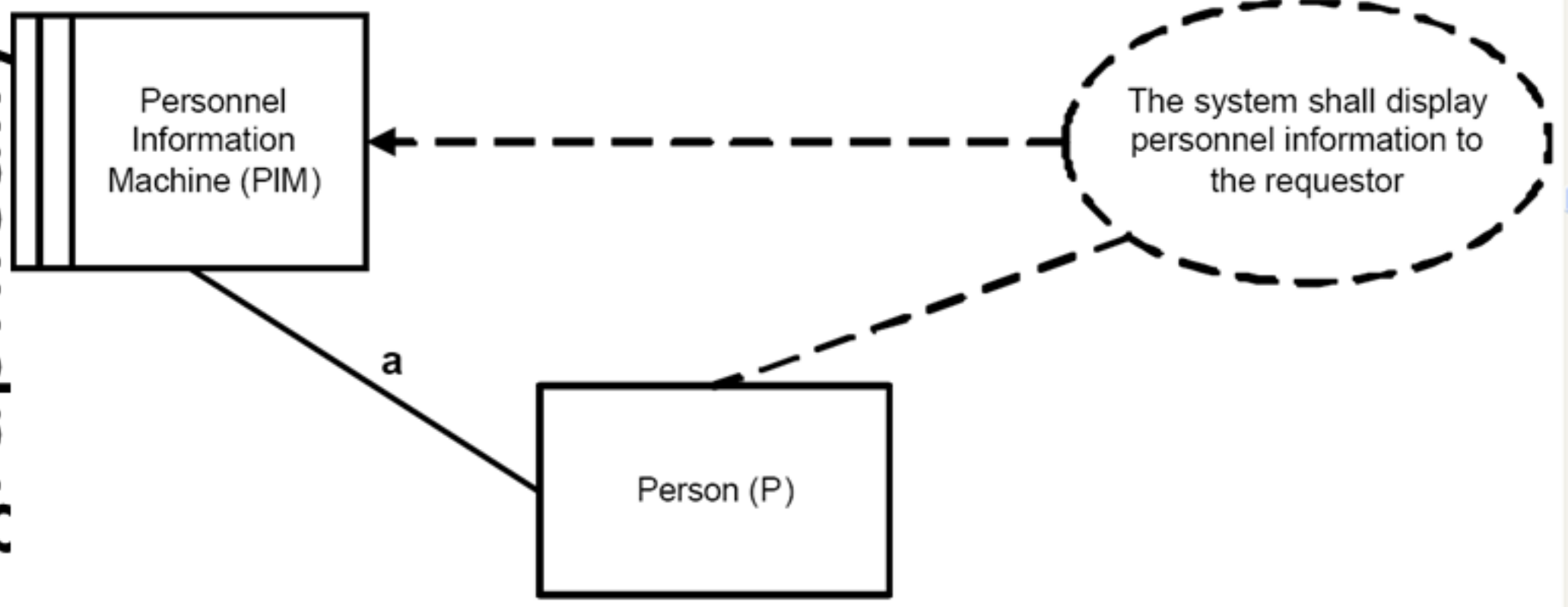


# Case Study

- Personal Information Display System
  - Unrealistically simple example
- System goal
  - Provision of people's personnel information to them. *→ Show personal data.*
- System functional requirement
  - REQ1: On request from a Person (member of people), the system shall display personnel information (PersonInf) for a specified payroll number (Payroll#) to that Person.

# Requirement shown

graphically



# Step 1: Asset identification

- What are the relevant assets to this problem?

Personal Info.

→ SSN

→ Salary

→  
⋮

What are the relevant harms?



# Relevant Harms

- H1. Unauthorized disclosure

⇒ Someone views data who shouldn't

- H2. Unauthorized alteration

⇒ Someone changes your data.

- H3. Unavailability

⇒ System is ~~down~~  
brought of-line.

# Security Goals

- SG1: Confidentiality: Prevent unauthorized disclosure of personnel information.
- SG2: Integrity: prevent unauthorized alteration of personal information.
- SG3: Availability: Ensure availability of personnel information.

# Translating Security Goals to Security Requirements

- Step 1
  - Examine goal for relevance —
  - Operationalize requirement by deriving constraints on the functional requirements

# SG1

- SG1: Confidentiality: Prevent unauthorized disclosure of personnel information.



# SG1

- SG1: Confidentiality: Prevent unauthorized disclosure of personnel information.
- REQ1/SR1: The machine shall display personnel information only to members of the HR department.
  - Only requirement for confidentiality.

# SG2

- SG2: Integrity: prevent unauthorized alteration of personal information.
- REQ1/SR1: The machine shall display personnel information only to members of the HR department.
  - Only requirement for confidentiality.

*N/A*

# Resulting Security Requirements Model

