Lecture 2

Security Threat Modeling Techniques

Dr. Charnon Pattiyanon
Assistant Director of IT, Instructor
Department of Artificial Intelligence and Computer Engineering
CMKL University

Today's Class Outline

- Upon successful of this lecture, you will know about:
  - The approaches and techniques to ***model security threats*** of an information system and the benefits on how they help system analysts to identify hidden security threats.
  - ***Basic security models*** that could be helpful to protect an information system against security threats.

Recap on What Are Security Threats

Security Threats!

- **Security threat (n.)** in computer engineering and science field is a set of harmful activities that could potentially threaten computer users or systems.

The Internet Engineering Task Force (IETF) defines "Security Threat" in **RFC 4545**[1] as "A potential for violation of security, which exists when there is an entity, circumstance, capability, action, or event that could cause harm".

The National Institute of Standardization and Technology (NIST) defines "Security Threat" in **NIST SP800-160, Vol. 2**[2] as "An event or condition that has the potential for causing asset loss and the undesirable consequences or impact from such loss".

Types of Security Threats

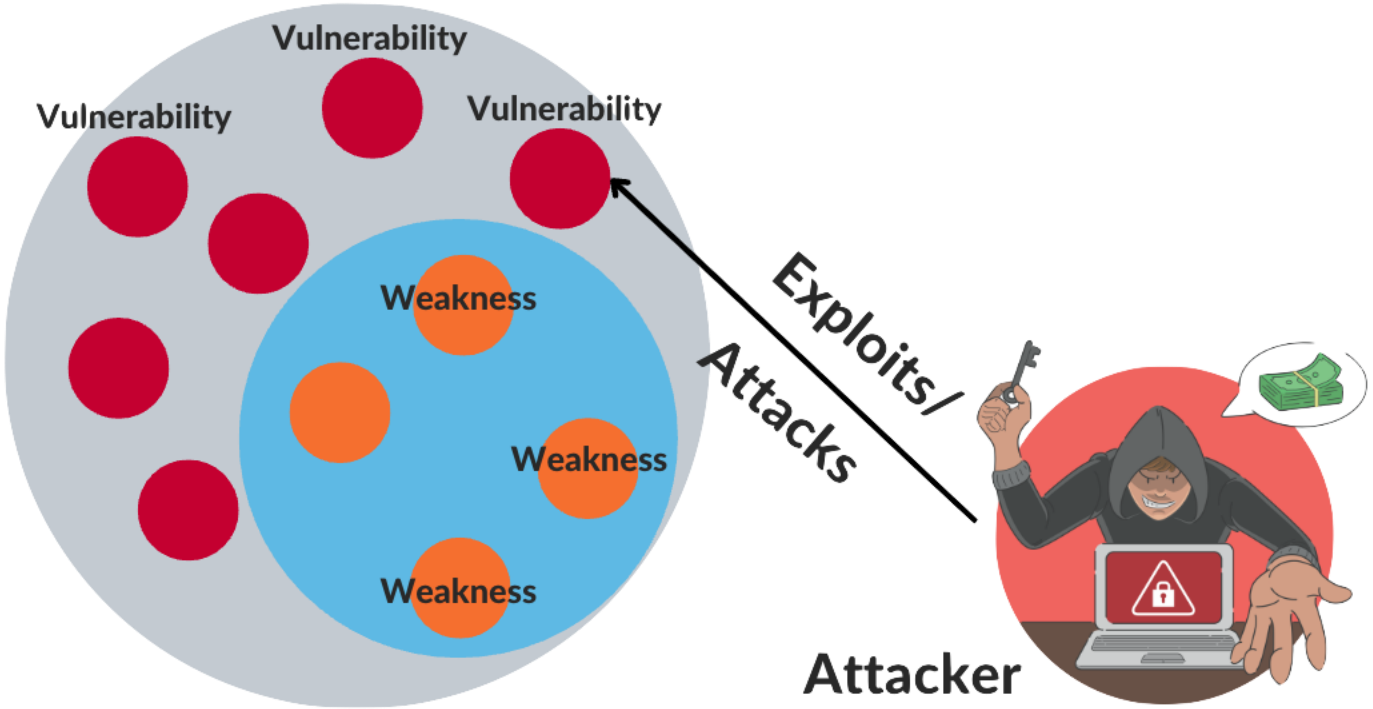
Threat	Meaning/Example	Related Security Property
Identity Spoofing	An attempt to use someone else's password to authenticate to services as that person.	Authentication
Data Tampering	An attempt to modify data either the data at rest, or the data during the trasmission.	Integrity
Repudiation	An attempt to deny having performed data manipulation.	Non-repudiation
Information Disclosure	An attempt to read user data without permission, or eavedrop a data communication channel.	Confidentiality
Denial of Service	An attempt to provide flood of traffic to a system with the intention to make it unavailable.	Availability
Elevation of Privilege	An attempt to gain higher privilege to a less privilege account.	Authorization

Weaknesses and Vulnerabilities Lead to Threats

- Security threats occurred when attackers exploit on system's weaknesses, or vulnerabilities to gain some benefits.

"A ***weakness*** is a condition in a software, firmware, hardware, or service component that, under certain circumstances, could contribute to the introduction of vulnerabilities." – ***MITRE foundation***.

"A ***vulnerability*** is a flaw in a software, firmware, hardware, or service component resulting from a weakness that can be exploited, causing a negative impact to the confidentiality, integrity, or availability of an impacted component or components." – ***MITRE foundation***.

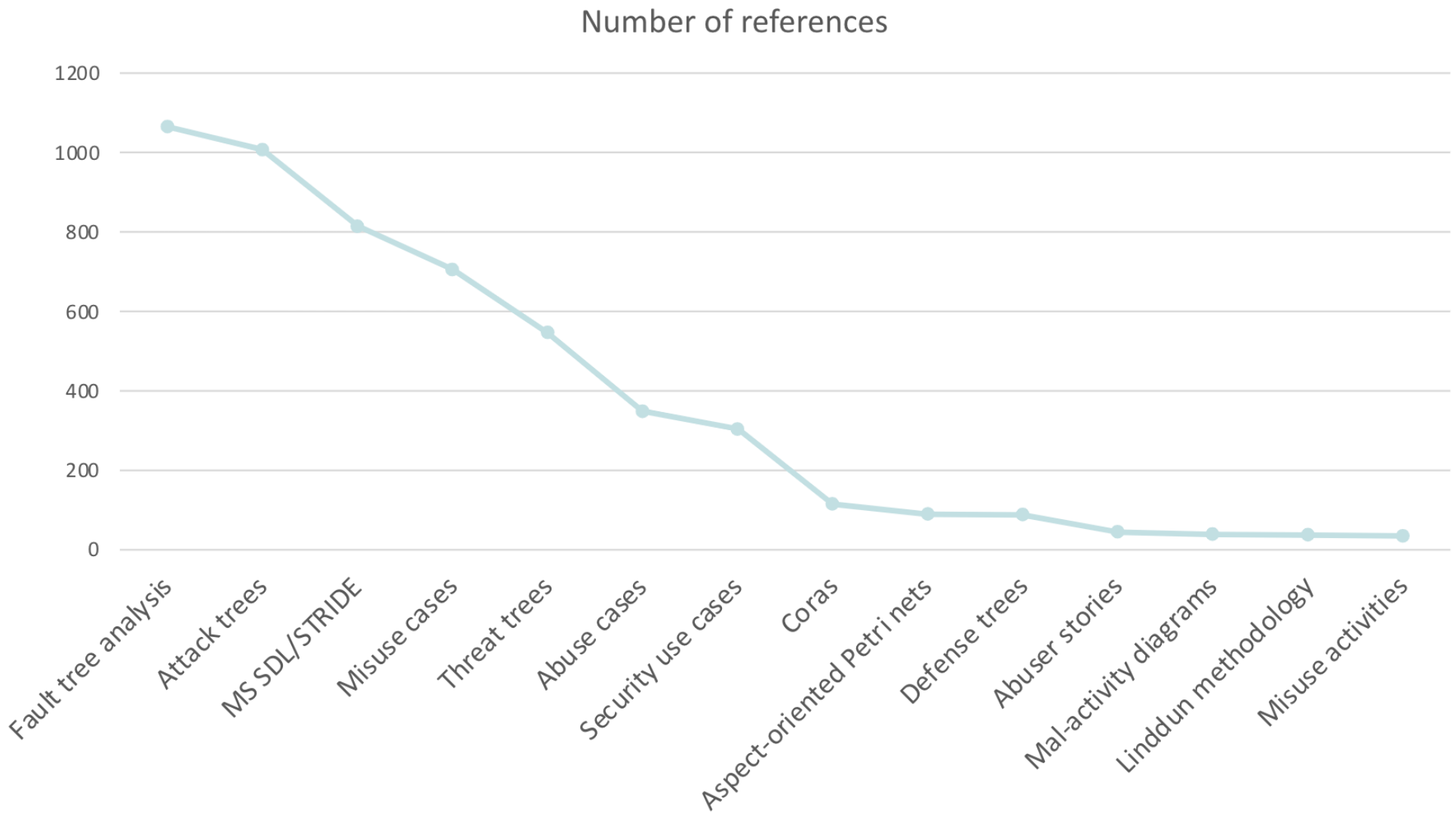


Security Threat Modeling Techniques

- ***Threat Modeling*** is a group of techniques or methods to **identify** potential threats to a target system.

- There are **three main approaches**, which are:
  - Attacker Approaches: This group of approaches aims to identify **who are the attacker**, what are their intentions to attack, and how they can exploit the system.
  - Asset Approaches: This group of approaches aims to **identify values of asset** in the target system, and how attackers can reach the asset.
  - Software Approaches: This group of approaches aims to **identify potential vulnerabilities** that attackers are likely to exploit in a software system, and how attackers can access to those vulnerabilities.

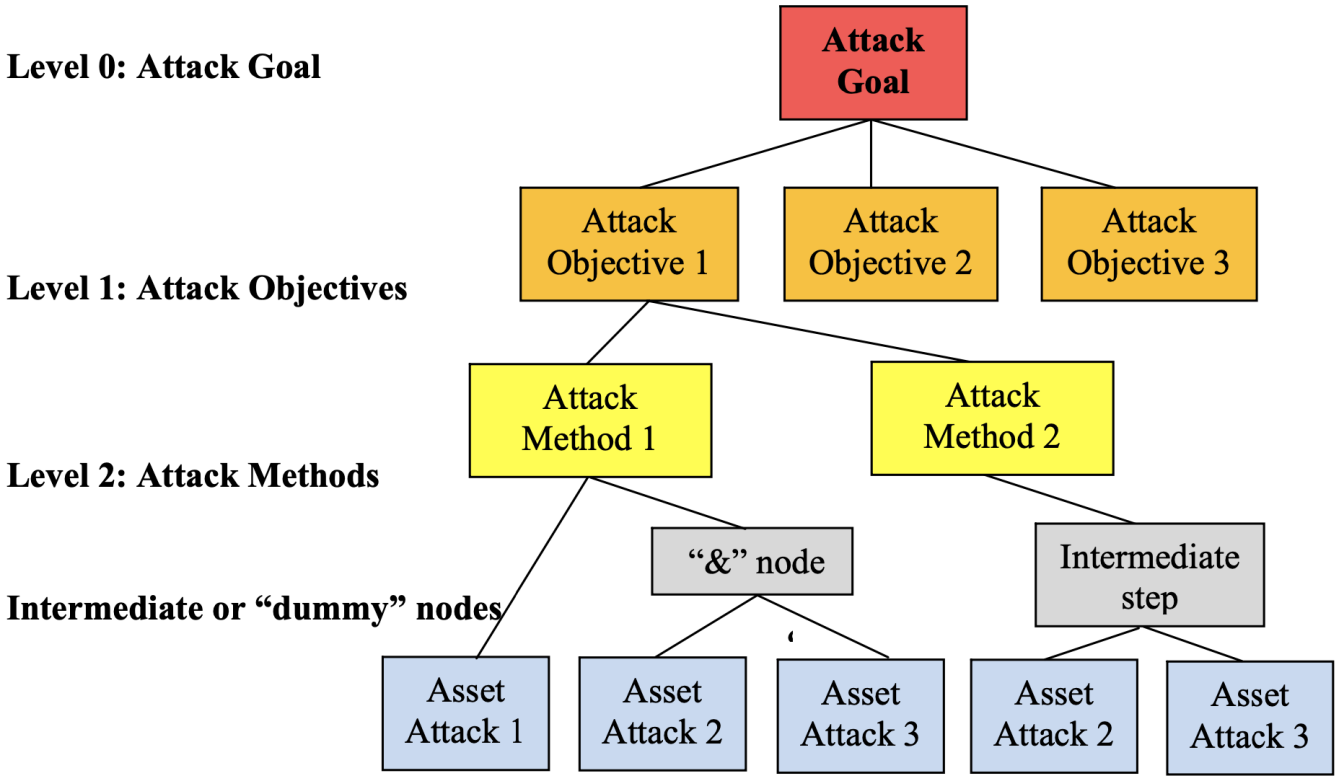
Security Threat Modeling Techniques (Cont')



Reference: A survey on the number of references using different threat analysis techniques [Türpe and Poller, 2015]

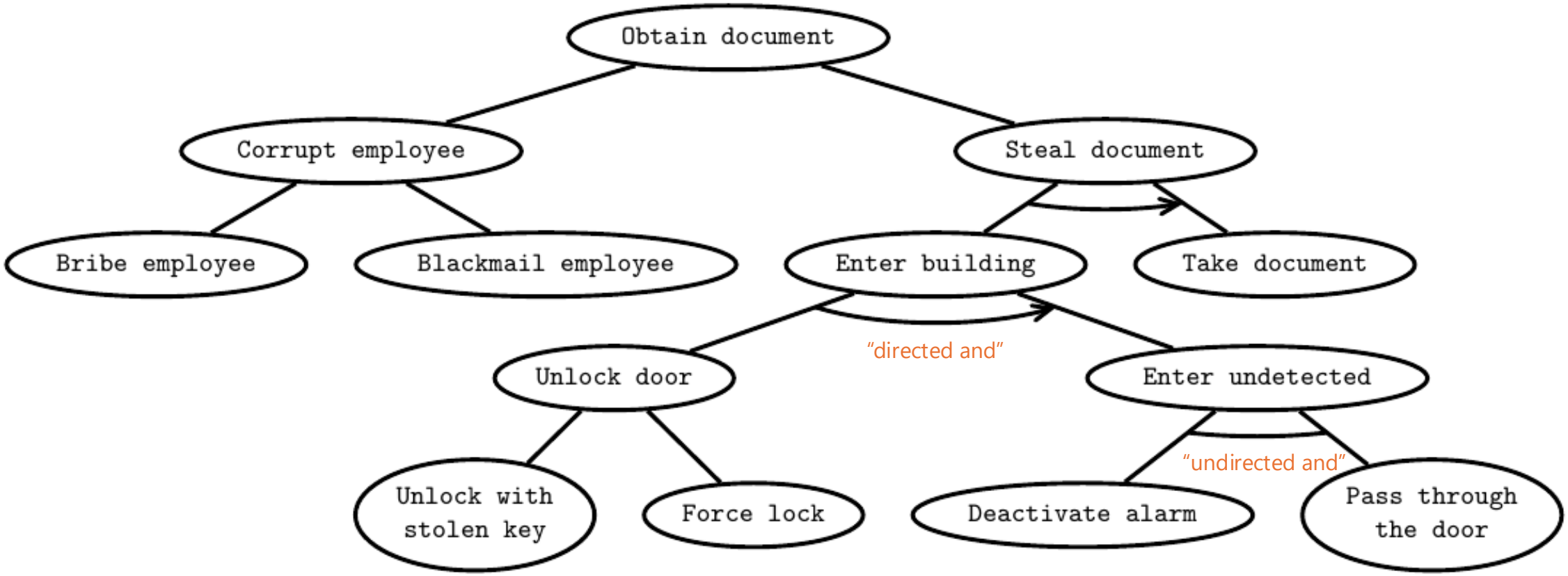
1. Attack Tree Analysis

- **Attack Tree Analysis** formulates potential attacks on the target system in a tree structure, where tree roots are *attack goals* along with their *objectives*, and leaf nodes are different *ways* to achieve those goals and objectives.



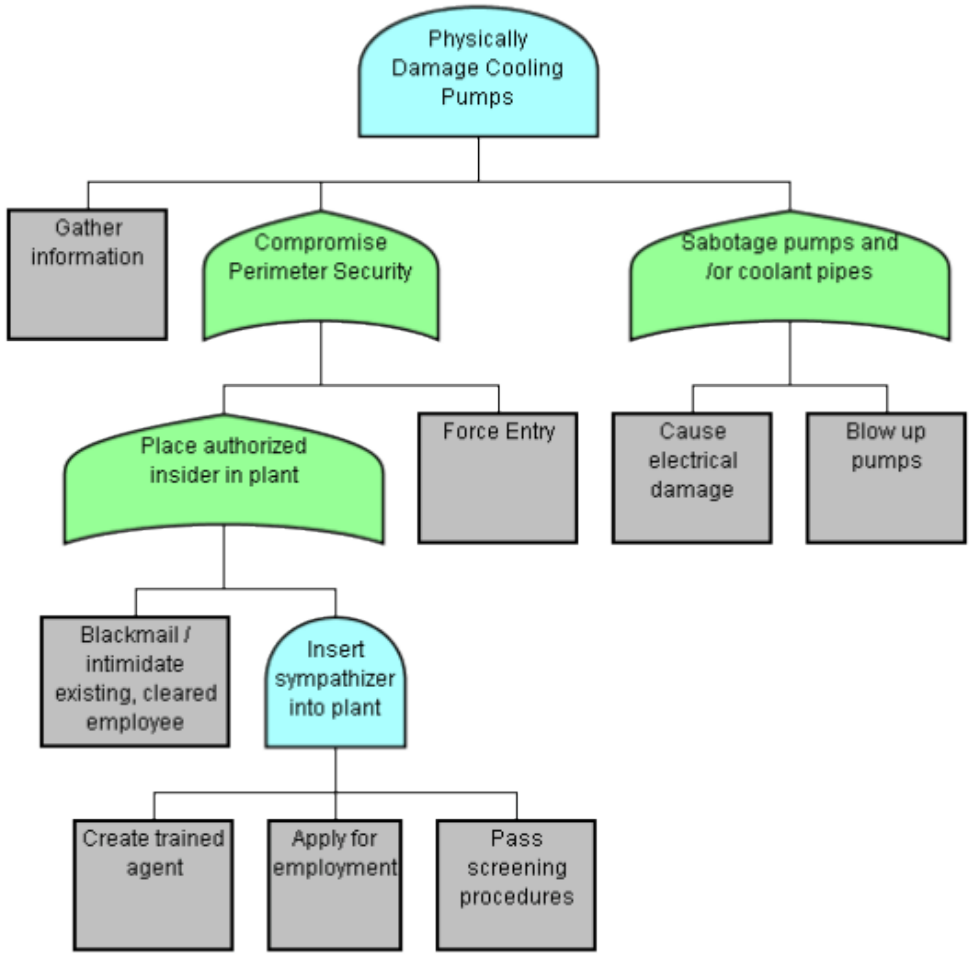
1. Attack Tree Analysis (Cont')

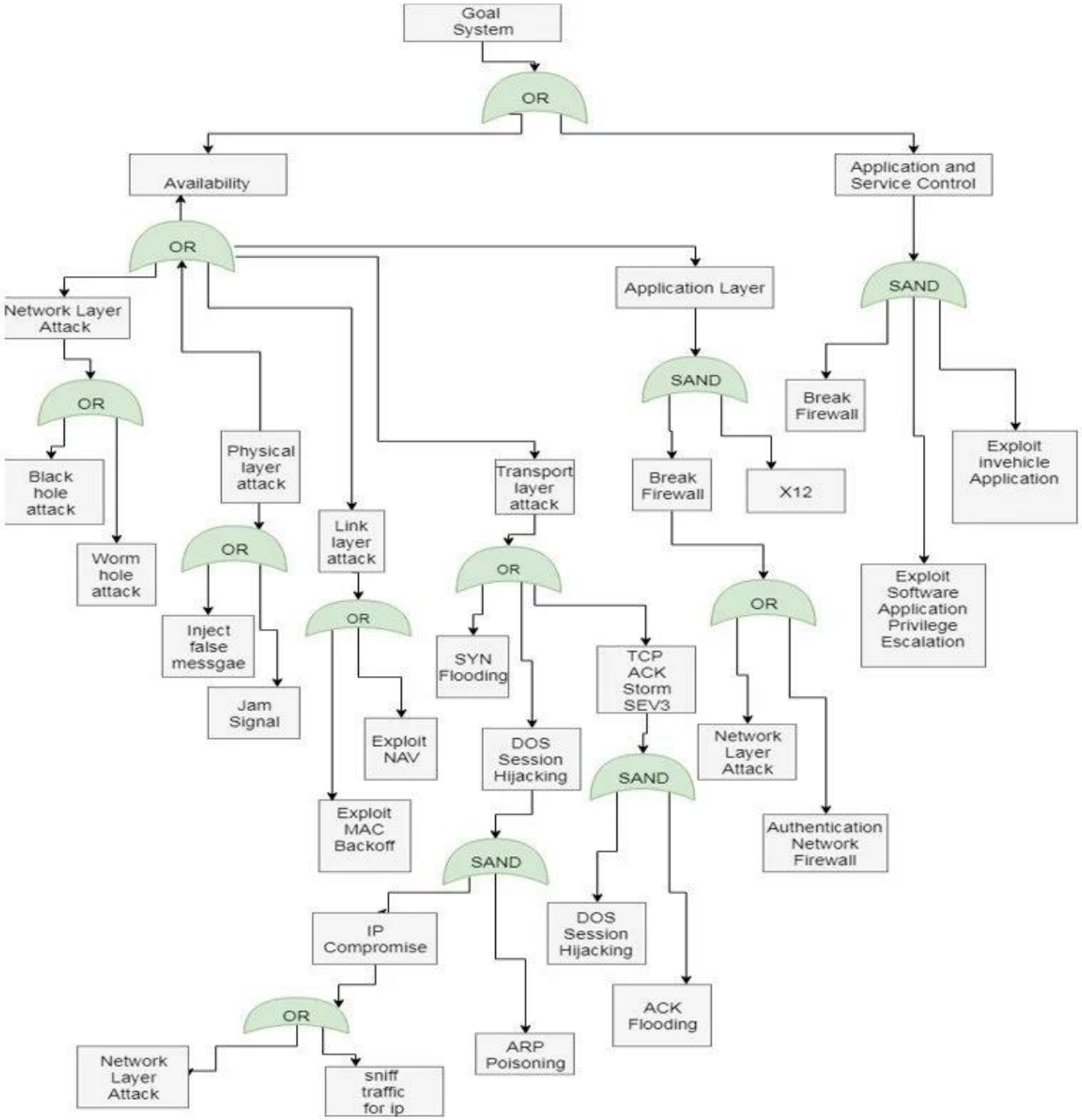
- **Examples of An Attack Tree Analysis Result:**



1. Attack Tree Analysis (Cont')

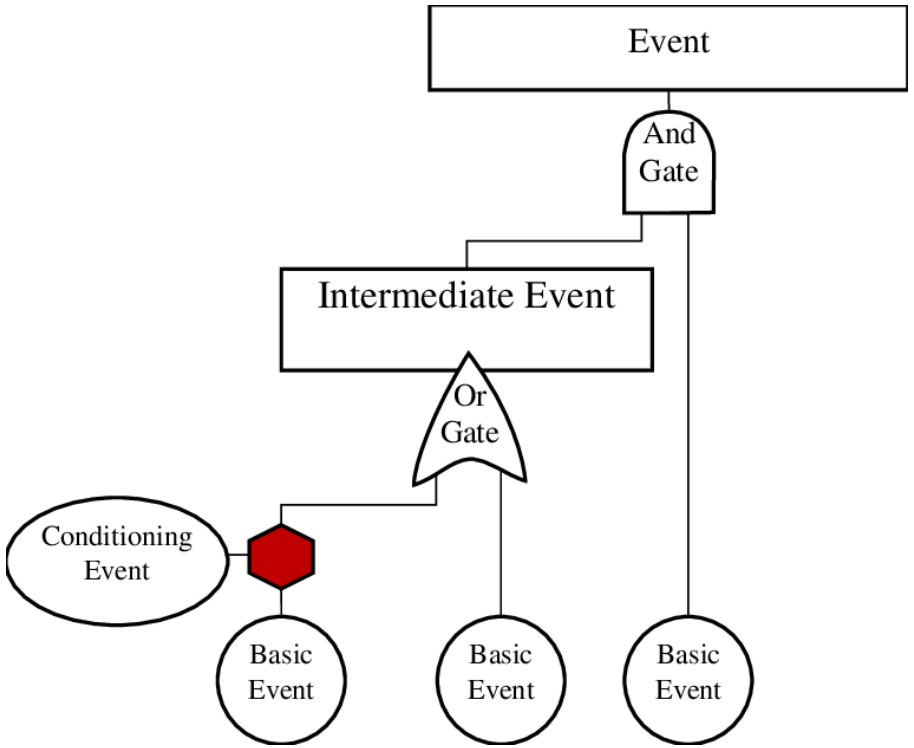
- **Examples of An Attack Tree Analysis Result:**

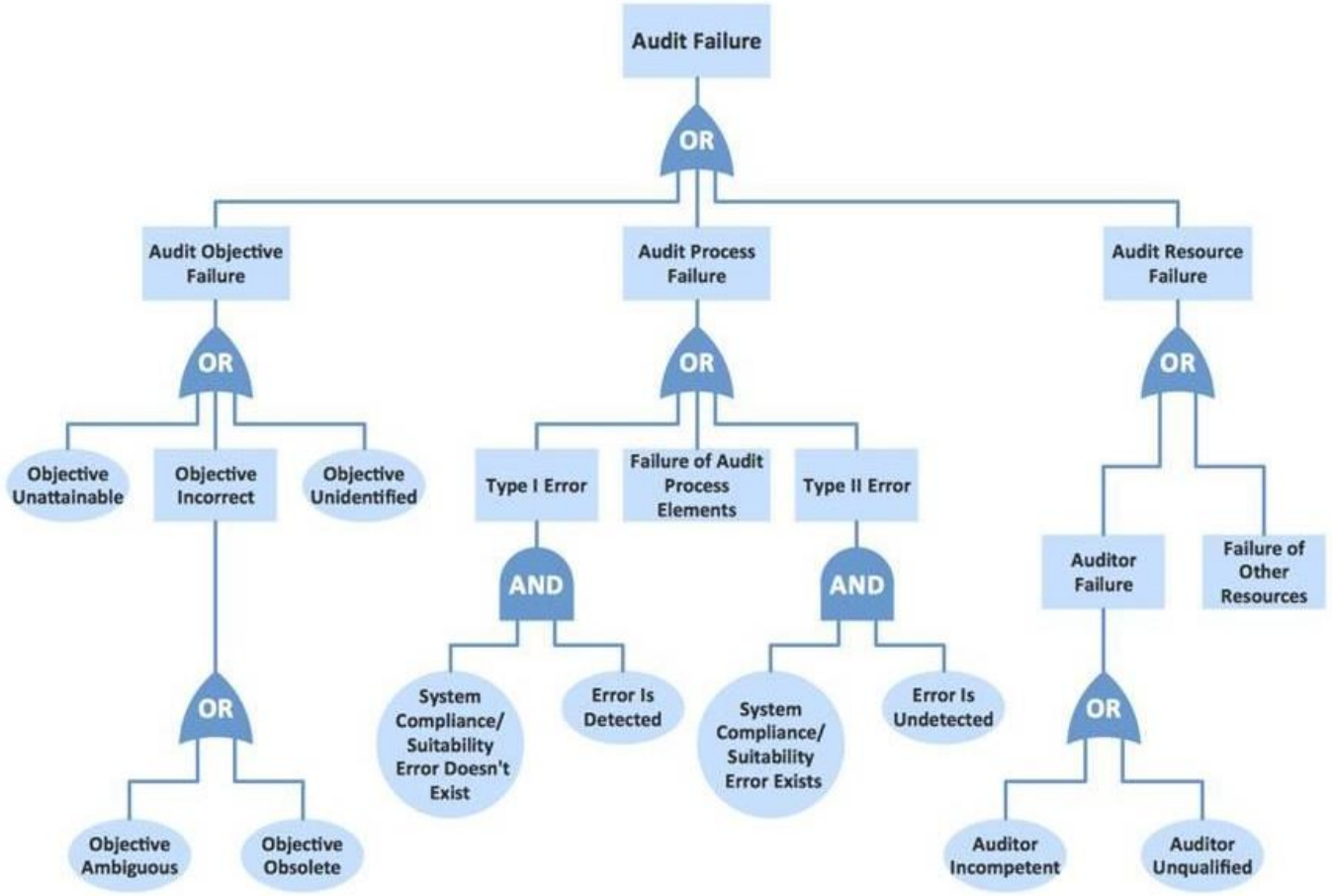




2. Fault Tree Analysis

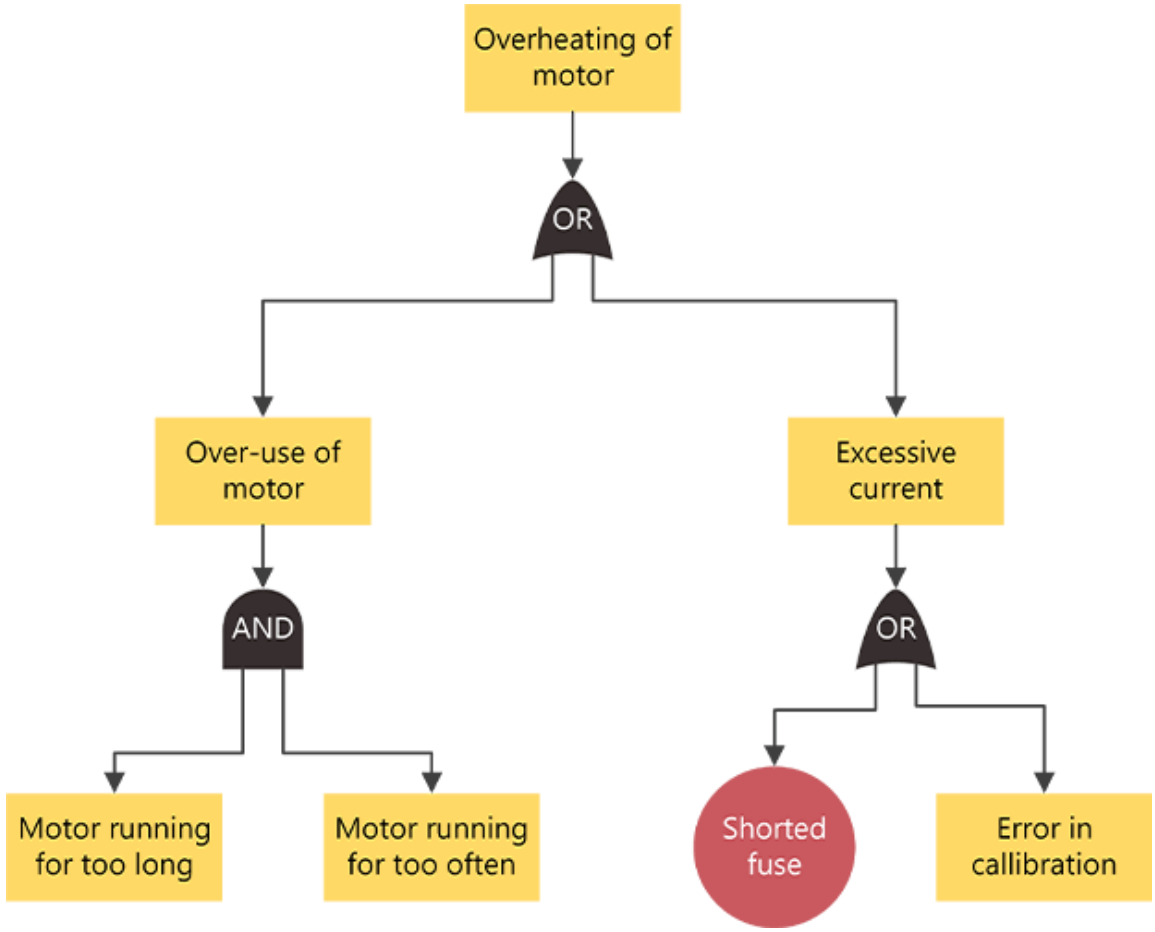
- **Fault Tree Analysis** is similar to the attack tree analysis, but it is a top-down approach to identify the potential *component-level failures* (***basic events***) that cause the *system-level failures* (***top events***) to occur.

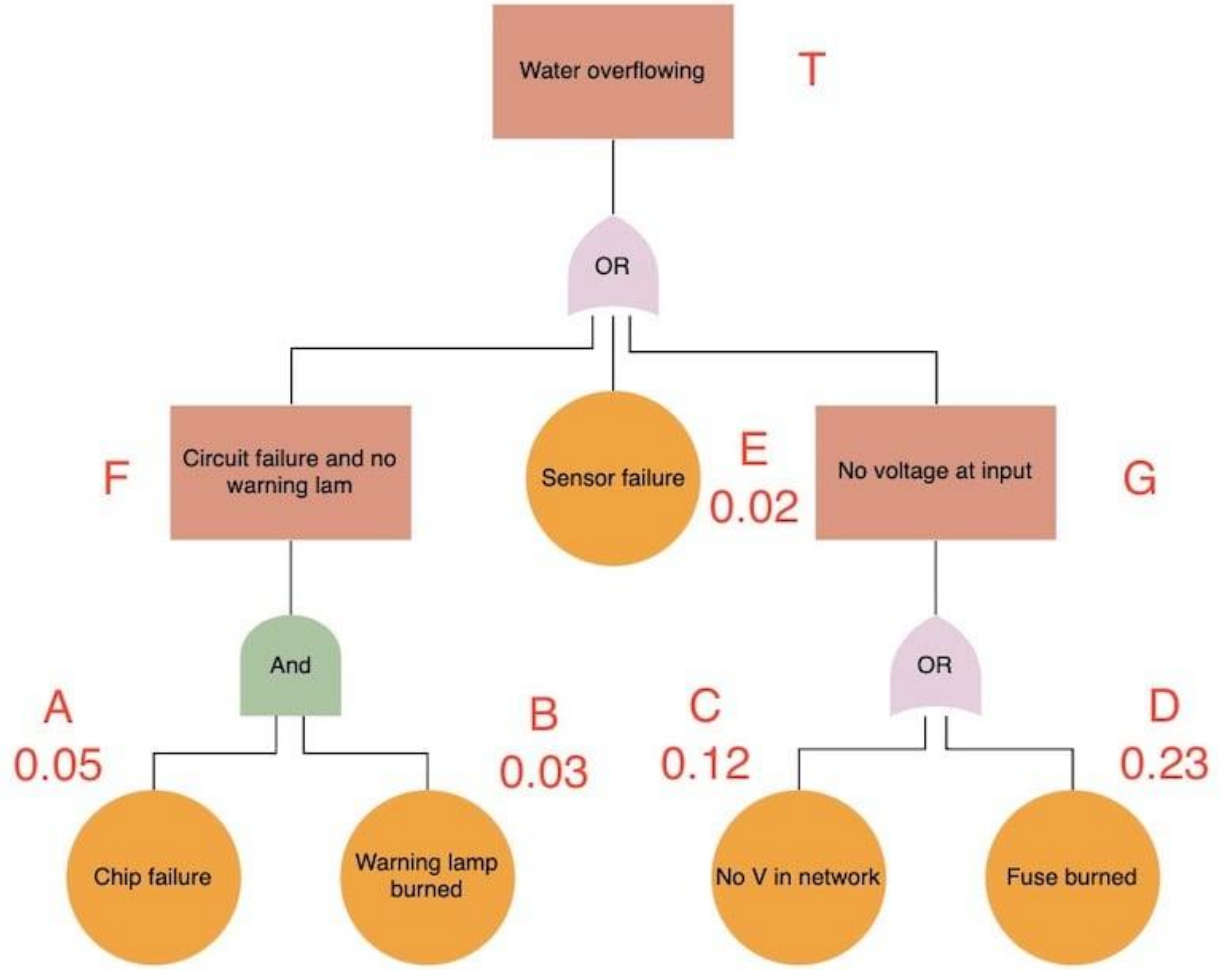




2. Fault Tree Analysis (Cont')

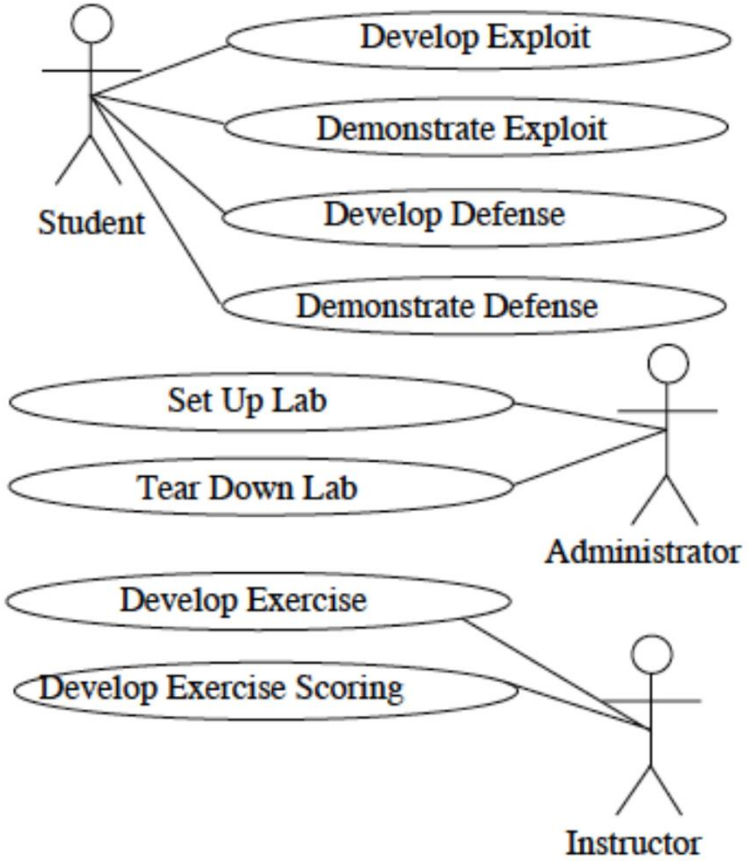
- **Examples of Fault Tree Analysis Results:**

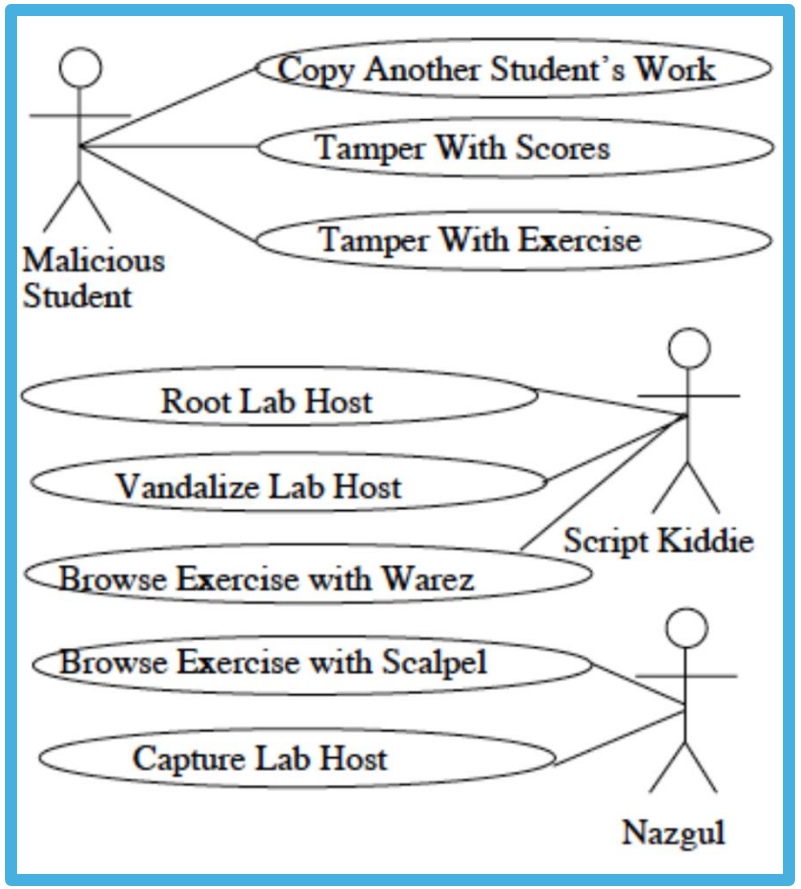




3. Abuse Case Analysis

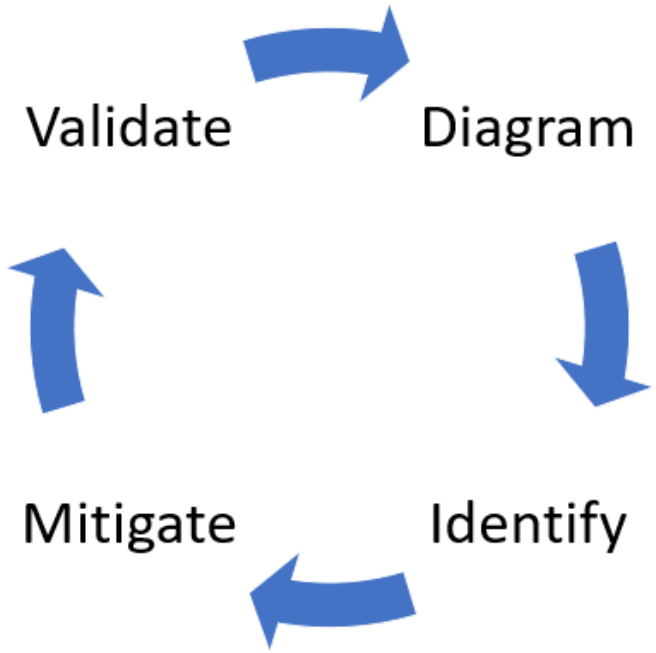
- **Abuse Case Analysis** is the opposite technique of *use case analysis* (i.e., modeling use cases that users will interact with the target system), which aims to ***identify cases that are considered abuse*** to the system.

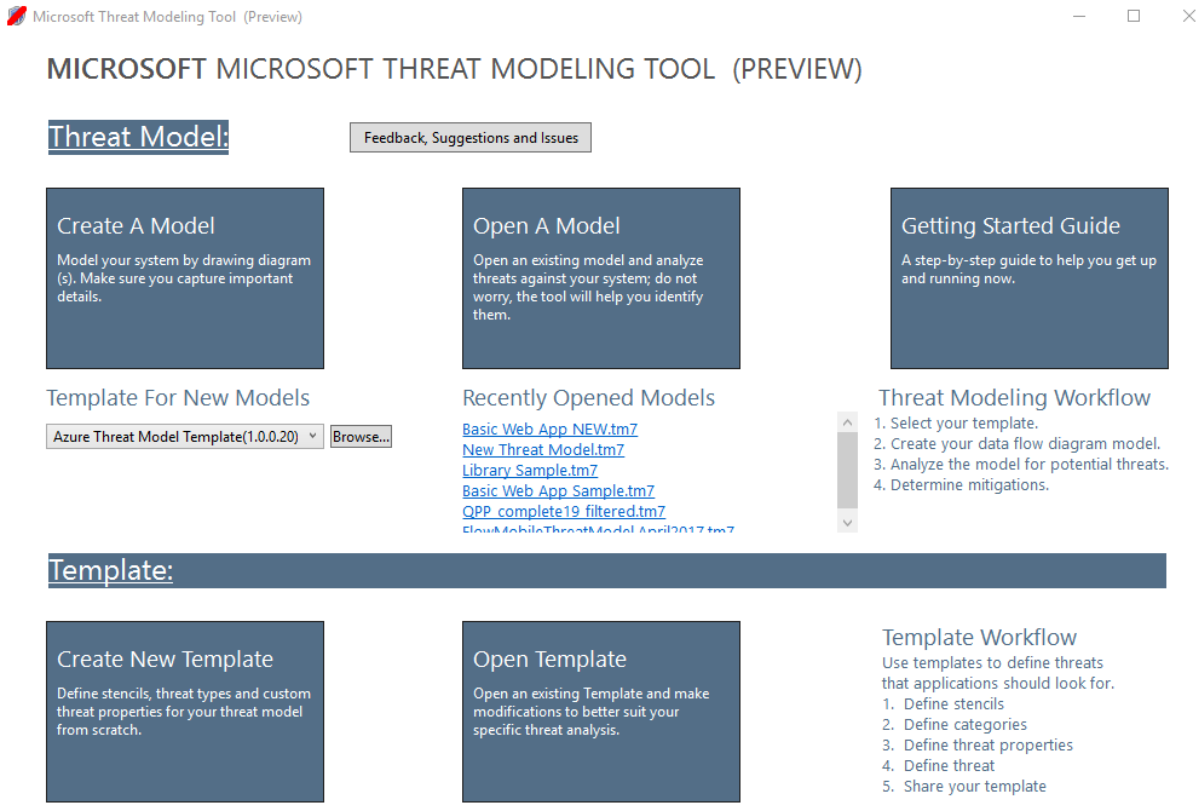




4. Microsoft Secure Development Lifecycle

- **Microsoft's Security Development Lifecycle (MS SDL)** ***Threat Modeling Tool*** is a well-known technique and set of tools introduced by Microsoft to help model target systems, analyze threats from predictable categories, and document them systematically.

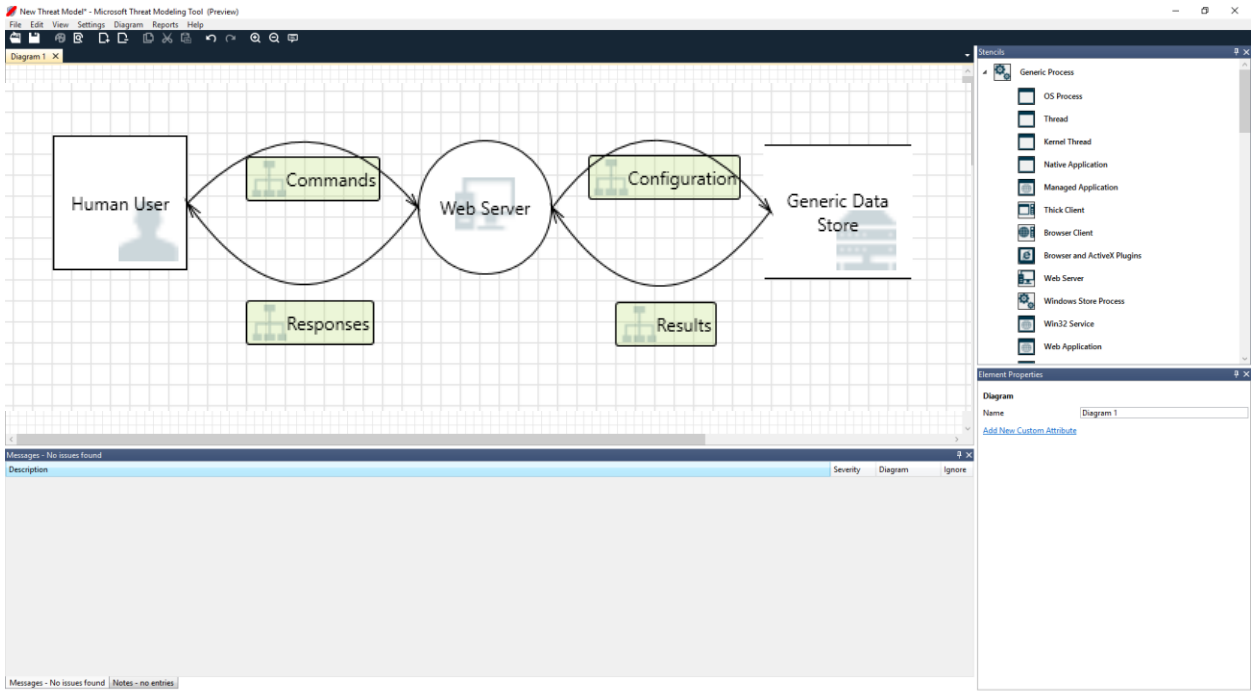


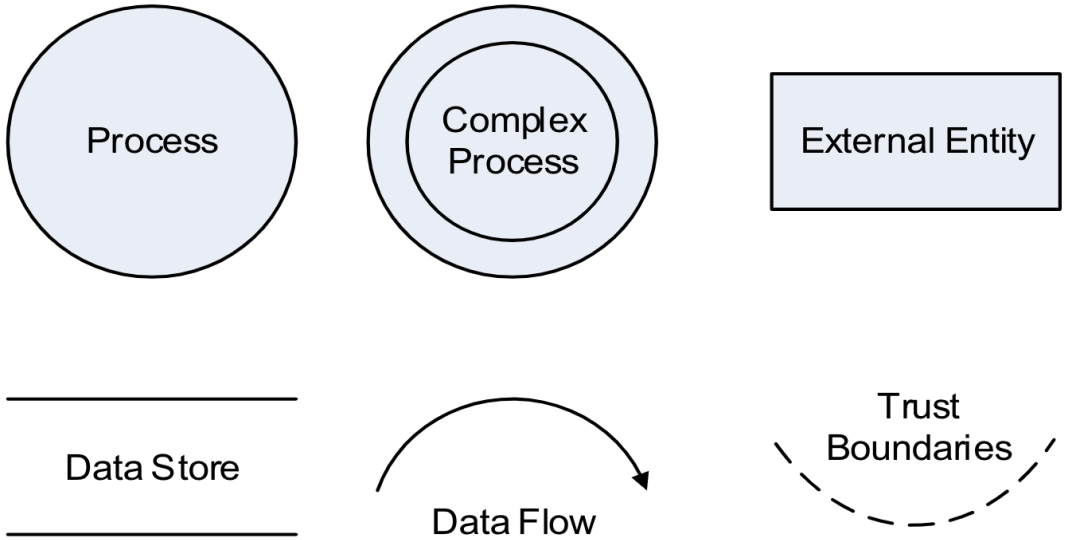


4. Microsoft Secure Development Lifecycle (Cont')

- **Microsoft's Security Development Lifecycle (MS SDL)** ***Threat Modeling Tool*** is a well-known technique and set of tools introduced by Microsoft to help model target systems, analyze threats from predictable categories, and document them systematically.

Step 1: ***Modeling the target system with a Data Flow Diagram (DFD)***

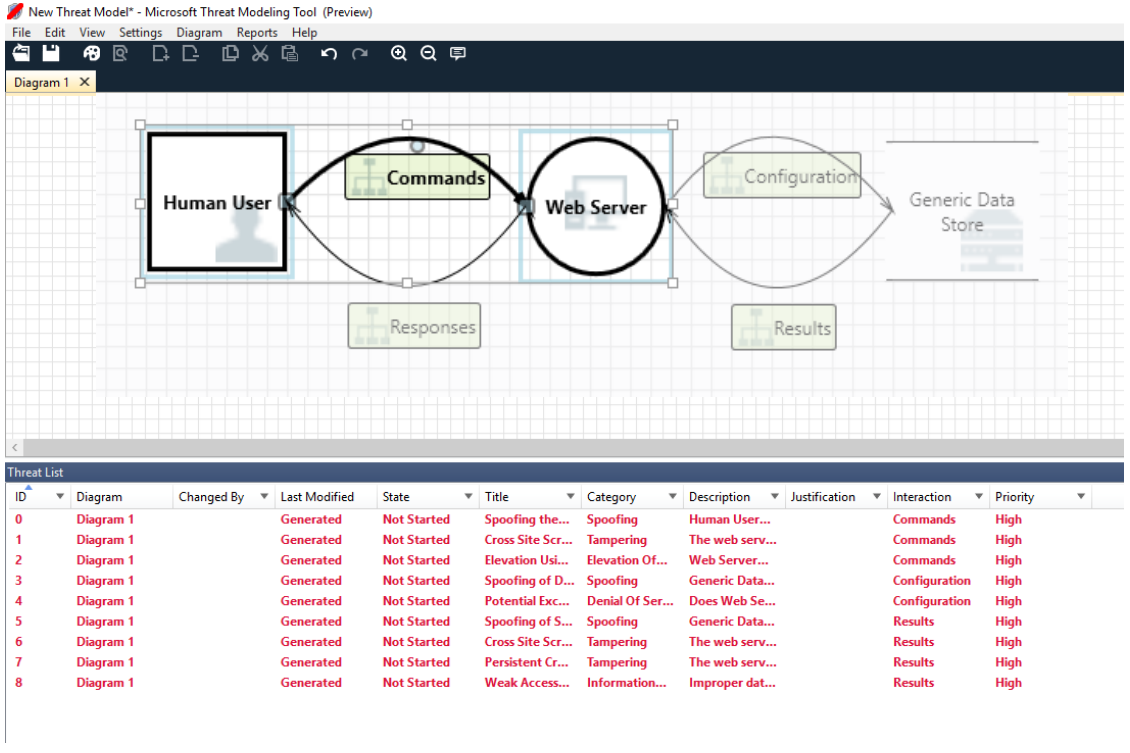




4. Microsoft Secure Development Lifecycle (Cont')

- **Microsoft's Security Development Lifecycle (MS SDL)** ***Threat Modeling Tool*** is a well-known technique and set of tools introduced by Microsoft to help model target systems, analyze threats from predictable categories, and document them systematically.

Step 2: ***The tool automatically analyzes the model against predefined sets of threats.***



The tool analyzes threats using the **STRIDE model**: **S**poofing, **T**ampering, **R**epudiation, **I**nformation Disclosure, **D**enial of Service, and **E**levation of Privilege.

DFD entity	S	T	R	I	D	E
External Entity	X		X			
Data Flow		X		X	X	
Data Store		X	(X)	X	X	
Process	X	X	X	X	X	X

4. Microsoft Secure Development Lifecycle (Cont')

- **Microsoft's Security Development Lifecycle (MS SDL)** ***Threat Modeling Tool*** is a well-known technique and set of tools introduced by Microsoft to help model target systems, analyze threats from predictable categories, and document them systematically.

Step 3: ***The tool automatically generates a threat modeling report.***

Threat Modeling Report

Created on 7/31/2017 12:35:42 PM
Threat Model Name:
Owner:
Reviewer:
Contributors:
Description:
Assumptions:
External Dependencies:

Threat Model Summary:

Not Started	9
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	0
Total	9
Total Migrated	0

Diagram: Diagram 1

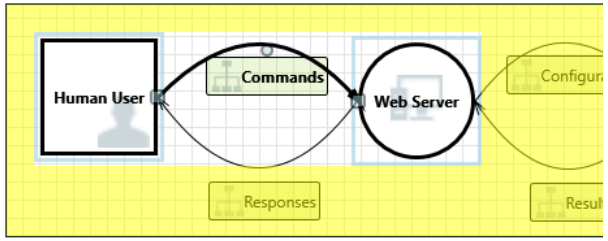
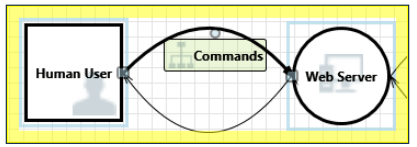


Diagram 1 Diagram Summary:

Not Started	9
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	0
Total	9
Total Migrated	0

Interaction: Commands



1. Spoofing the Human User External Entity [State: Not Started] [Priority: High]

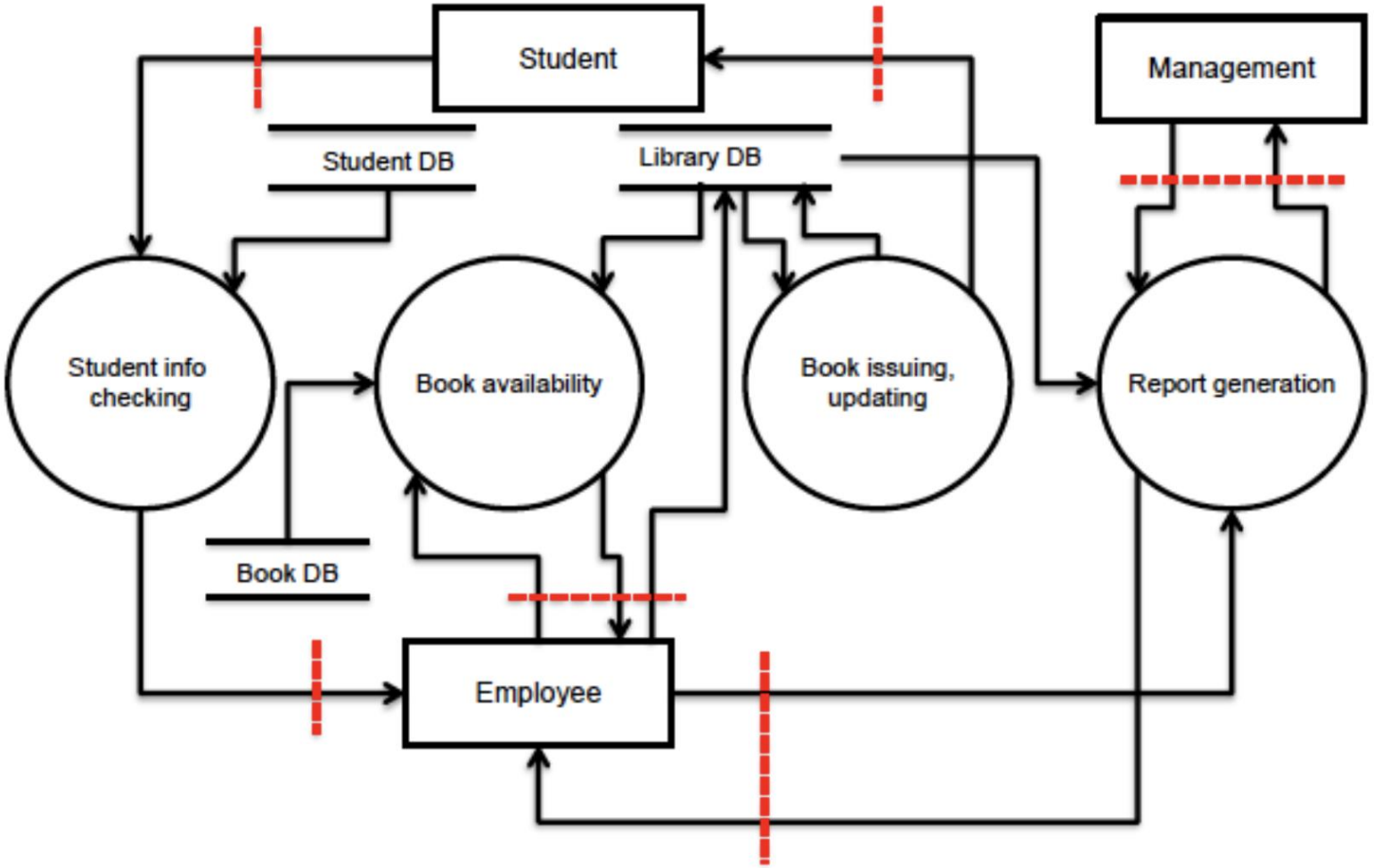
Category:	Spoofing
Description:	Human User may be spoofed by an attacker and this may lead to unauthorized access to Web Server. Consider using a standard authentication mechanism to identify the external entity.
Justification:	<no mitigation provided>
Possible Mitigation(s):	
SDL Phase:	Design

2. Cross Site Scripting [State: Not Started] [Priority: High]

Category:	Tampering
Description:	The web server 'Web Server' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Justification:	<no mitigation provided>
Possible Mitigation(s):	
SDL Phase:	Design

4. Microsoft Secure Development Lifecycle (Cont')

- **An example of DFD**, following the MS SDL's threat modeling tool.



5. Taint Analysis Techniques

- A commonly used tool for software security analysis, introduced by Funnywei (2003).
- This technique tracks the flow of data throughout a program, including its manipulation processes.
- Taint analysis operates in a tripartite manner, involving *a source*, *a sink*, and *a sanitizer*.

- **In a nutshell:**
  - Data from *sensitive* sources (e.g., private or potentially malicious inputs) is initially marked as **tainted**.
  - Any other data influenced by this tainted data also becomes tainted.
    - For example, if x is a user input, then in $y = x + 2$, the variable y is also considered tainted.

5. Taint Analysis Techniques (Cont')

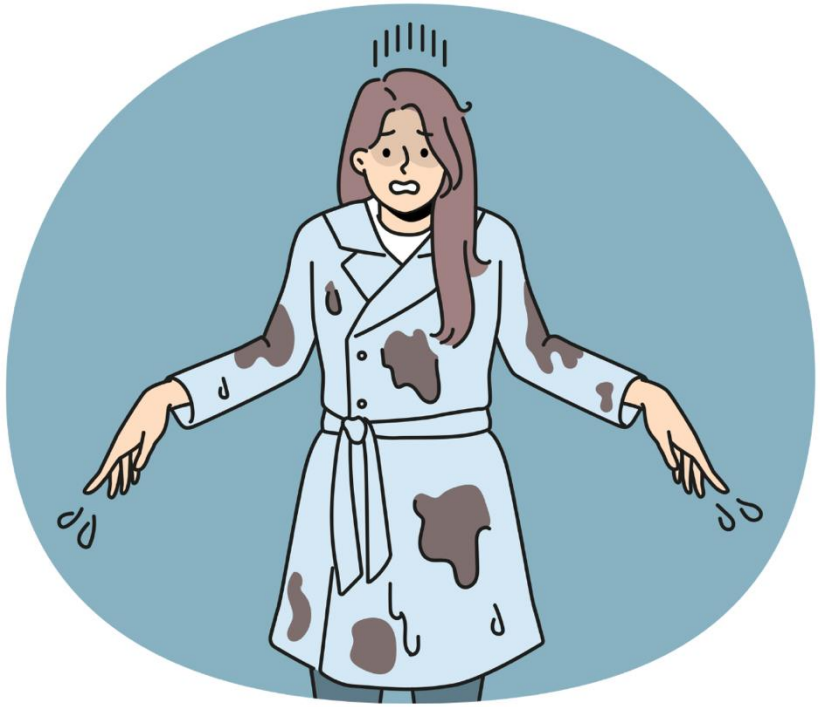
- There are **two types** of taint analysis techniques: (1) Static Taint Analysis, and (2) Dynamic Taint Analysis

Static Taint Analysis	Dynamic Taint Analysis

• Examine software (e.g., its intermediate code) or system **without** the actual execution.
• For software and applications, STA is mostly carried out in **two-step** manner:
  • Disassembly the intermediate code.
  • Conducting analysis on the resulting assembly code.
• The STA is difficult to combat malicious programs or systems with strong evasion techniques.
	• Examine software (e.g., its intermediate code) or system **during** the actual execution in the live environment.
	• Conducting vulnerability analysis on software in cases where **the source code is not available**, e.g., Commercial Off The Shelf (COTS), the DTA approach is the ideal option.
	• The DTA approach is less practical for the **system** security analysis.

5. Taint Analysis Techniques (Cont')

- How to conduct Dynamic Taint Analysis (DTA) or set a taint tracking policy:
  1. Identify sources of taint - "**Taint Seed**"
     - What are we going to track?
       - **Untrusted input**
       - **Sensitive Data**
  2. Specify taint policies - "**Taint Tracker**"
     - Specify **methods** or **criteria** for taint **propagations**.
  3. Identify taint sinks - "**Taint Assert**"
     - How do we consider the taint to be exploited?
       - Special call (e.g., `jump` or `goto` statement), outside network, or external components.



5. Taint Analysis Techniques (Cont')

- **How to conduct Dynamic Taint Analysis (DTA) or set taint tracking policy:**
  - **Example of a Taint Policy:**
    - Taint Seed: Any input from an untrusted source is tainted.

$$Input \frac{t = IsUntrusted(Src)}{getInput(src) \downarrow t}$$

    - Taint Tracker: For a binary operation, the taint of the result is the OR of the taints of each operator input.

$$BinOp \frac{t_1 = \tau[x_1], t_2 = \tau[x_2]}{x_1 + x_2 \downarrow t_1 \vee t_2}$$

    - Taint Assert: Any `goto` statement can only go to an untainted address.

$$P_{goto}(t_a) = \neg t_a$$
(Must be true to execute)

5. Taint Analysis Techniques (Cont')

- **How to conduct Dynamic Taint Analysis (DTA) or set taint tracking policy:**

```
x = get_input(    )

y = x + 42

...

goto y
```

Taint Seed

$$Input\ \frac{t = IsUntrusted(Src)}{getInput(src) \downarrow t}$$

Δ

Variable	Value

τ

Variable	Tainted?

5. Taint Analysis Techniques (Cont')

- **How to conduct Dynamic Taint Analysis (DTA) or set taint tracking policy:**

```
x = get_input(
y = x + 42
...
goto y
```

= Tainted

The input source is untrusted and the input data is tainted.

Δ

Variable	Value
x	7

τ

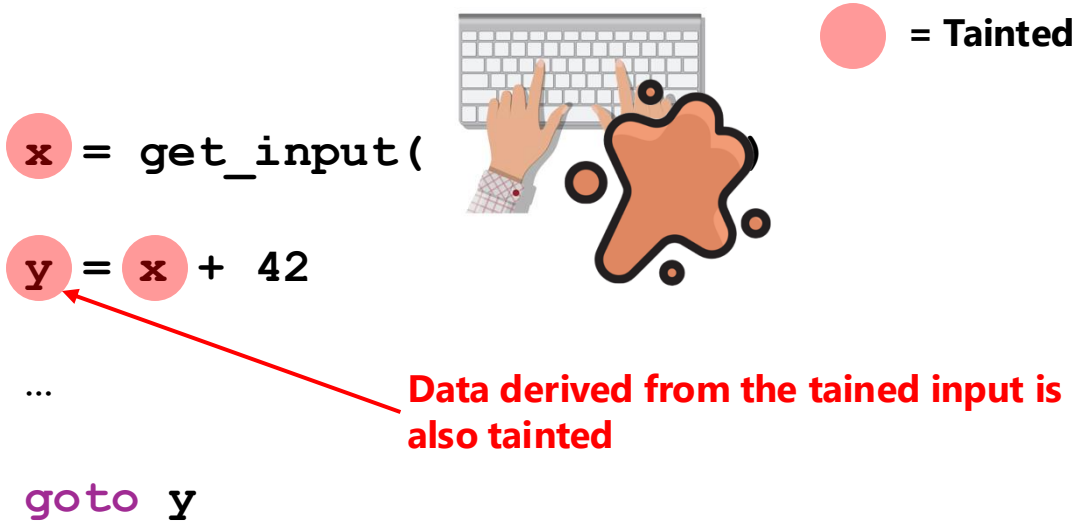
Variable	Tainted?
x	True

Taint Seed

$$Input\ \frac{t = IsUntrusted(Src)}{getInput(src) \downarrow t}$$

5. Taint Analysis Techniques (Cont')

- **How to conduct Dynamic Taint Analysis (DTA) or set taint tracking policy:**



Taint Tracker

$$BinOp\frac{t_1 = \tau[x_1], t_2 = \tau[x_2]}{x_1 + x_2 \downarrow t_1 \vee t_2}$$

Δ

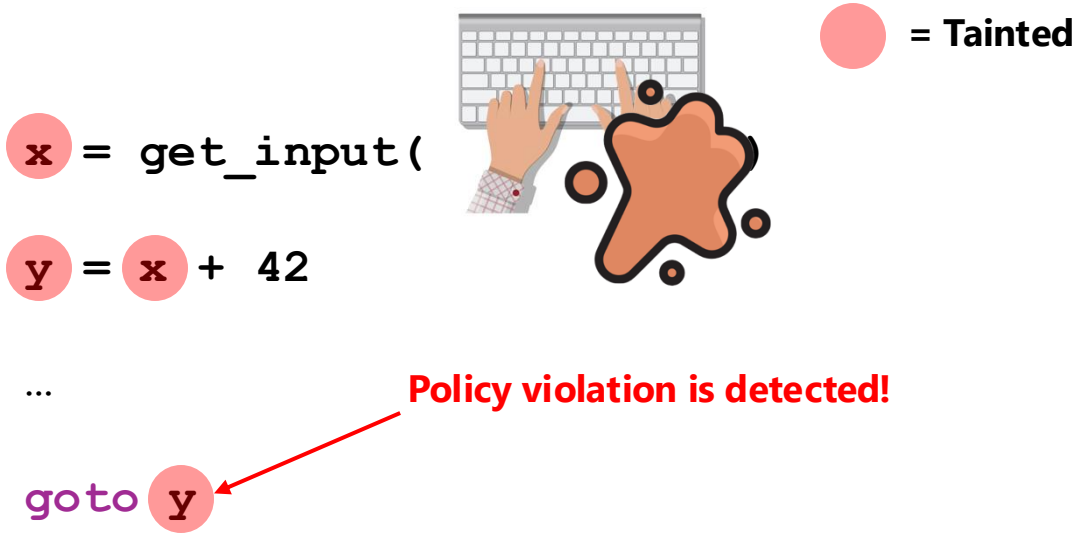
Variable	Value
x	7
y	49

τ

Variable	Tainted?
x	True
y	True

5. Taint Analysis Techniques (Cont')

- **How to conduct Dynamic Taint Analysis (DTA) or set taint tracking policy:**



$P_{goto}(t_a) = \neg t_a$
(Must be true to execute)

Δ

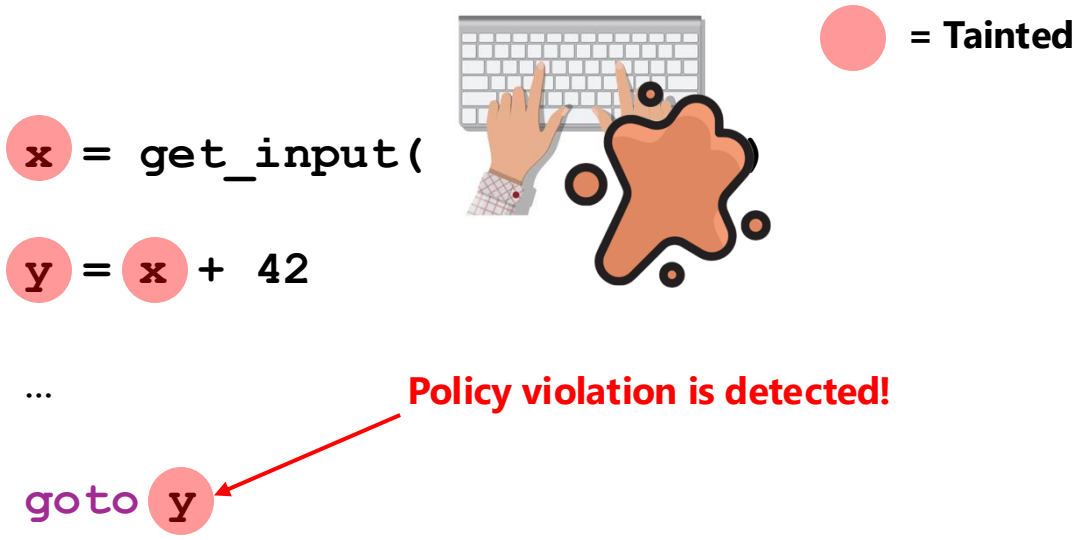
Variable	Value
x	7
y	49

τ

Variable	Tainted?
x	True
y	True

5. Taint Analysis Techniques (Cont')

- **How to conduct Dynamic Taint Analysis (DTA) or set taint tracking policy:**



In summary, this source code is vulnerable since it allows an input from an untrusted source to propagate to the special call statement

This kind of policy can be helpful for detecting buffer overflows:

```
...
strcpy(buffer, argv[1])
...
return;
```

5. Taint Analysis Techniques (Cont')

- **Cautions for Conducting Taint Analysis:**
  - **False Negative:**
    - Beware of a situation when a tainted variable is used to ***change a control flow***.

      Example:

```
if(x == 0) y = 0; else if(x == 1) y = 1;
```

      (This situation is equivalent to x = y)
    - ***Tainted index into a hardcoded table:*** the value translation is not tainted by policy.
    - It is difficult to ***enumerating all sources of taint***.
  - **False Positives:**
    - It is possible that the taint analysis can report a lot of tainted data and some of them are not actually tainted.

Key	Value
1	Jan
…	…

		Actual: True	Actual: False
Prediction	True	TP	FP
	False	FN	TN

A Summary of Threat Modeling Techniques

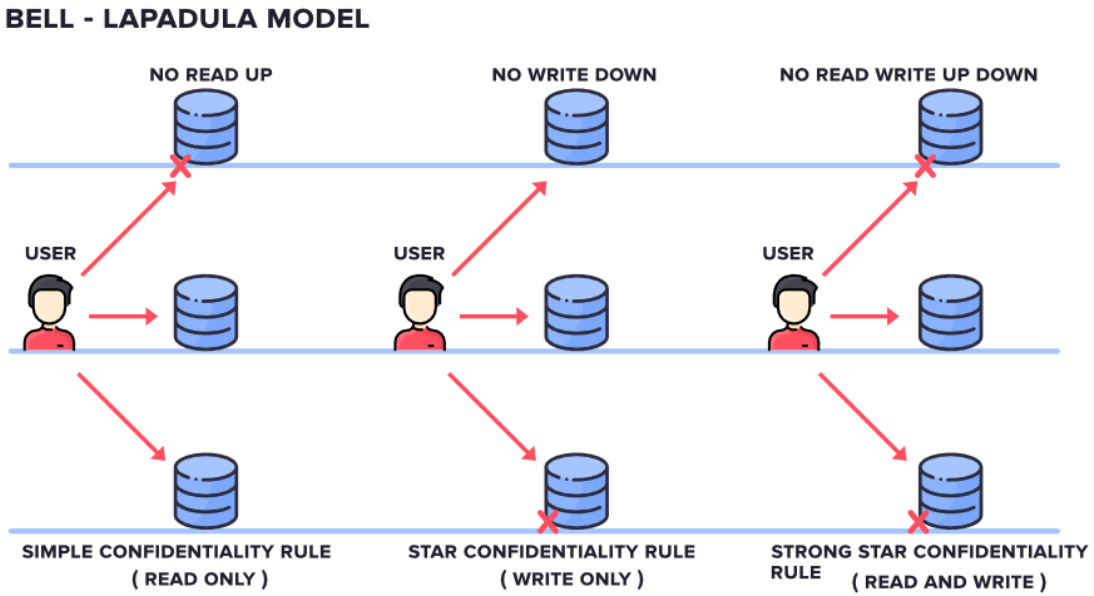
- Threat modeling techniques are aimed to **identify hidden security threats** that potentially occur in a target system.
- The main target of security threats are the ***system*** itself, ***assets***, or ***software*** application.
- The key expected outcomes of security threats are
  1. **Denial of Services** – Make the target system unavailable or failure
  2. **Access to unauthorized parts** – Make benefits from assets or information that should not be accessible.
  3. **Takeover the system or cause harmful events** – Make chaotic activities that cause harmful to the system's users or information.
- Threat modeling techniques did not protect security threats directly. So, how to protect?

Basic Security Models

- Security models are functional, protocol, or process models that are defined to ensure the alignment with security goals.
- There are found classic security models that are typically discussed:
    1. ***Bell-LaPadula Security Model***
    2. ***Biba Security Model***
    3. ***Clarke Wilson Security Model***
    4. ***Brewer-Nash Security Model***

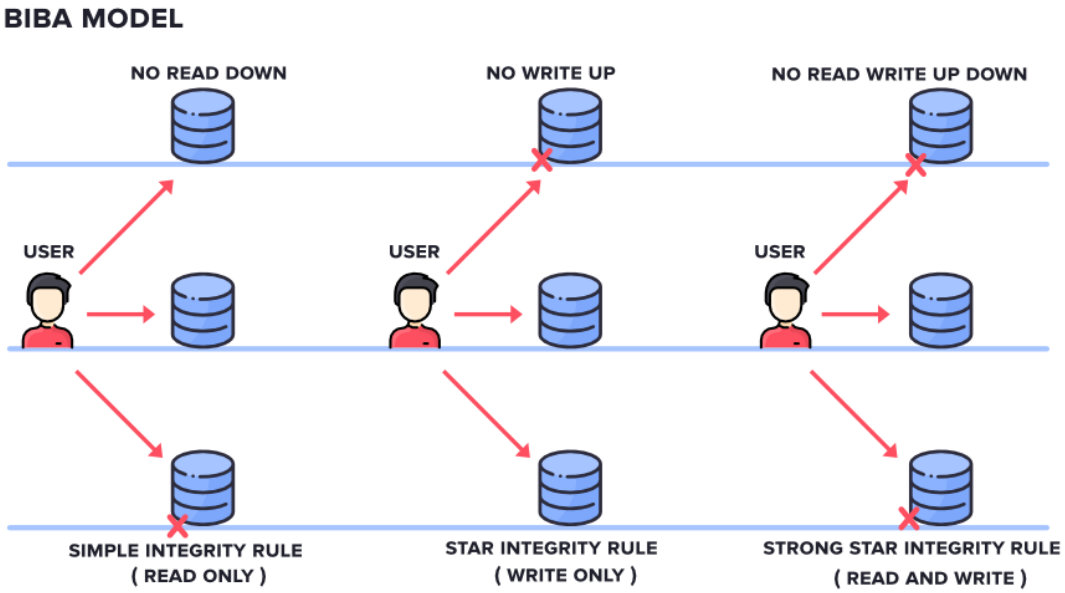
1. Bell-Lapadula Model

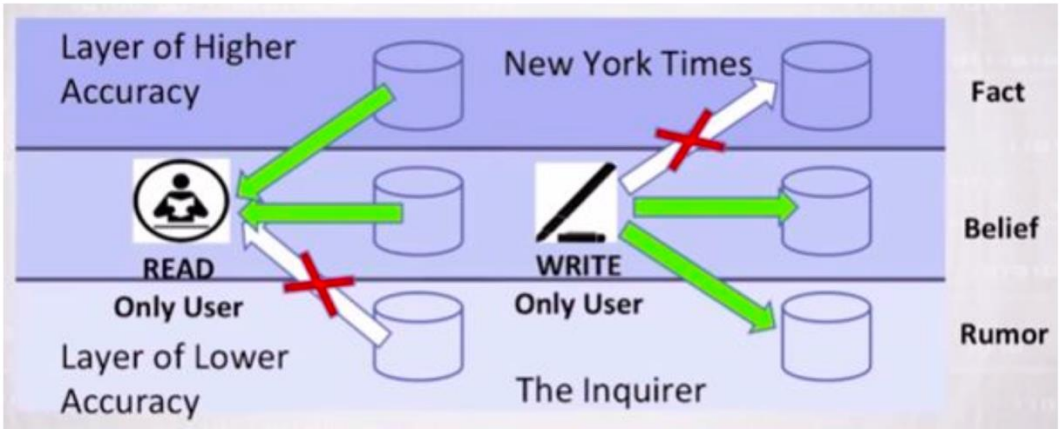
- **Bell-Lapadula Security Model** was developed by two scientists, David Elliot Bell, and Leonard J. LaPadula, to help maintaining the ***confidentiality*** of information.
- There are **three important rules**:
  - Simple Confidentiality Rule (***No Read Up***), Star Confidentiality Rule (***No Write Down***), and Strong Star Confidentiality Rule (***No Read Write Up Down***)



2. Biba Model

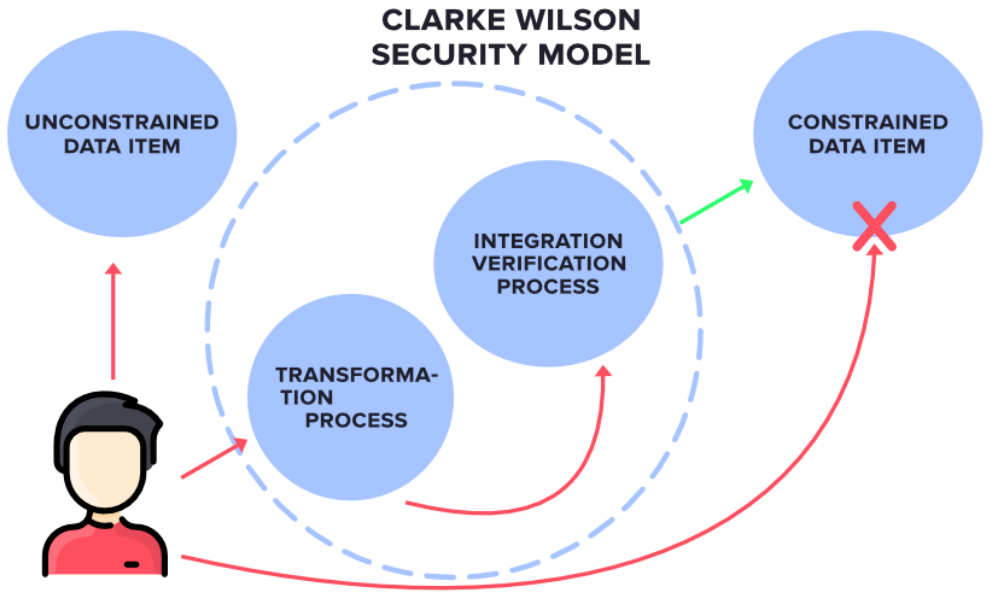
- **Biba Security Model** was developed by a scientist, Kenneth J. Biba, to help maintaining the ***integrity*** of information. It works the *exact reverse of Bell-LaPadula* security model.
- There are **three important rules**:
  - Simple Integrity Rule (***No Read Down***), Star Integrity Rule (***No Write Up***), and Strong Star Integrity Rule (***No Read Write Up Down***)





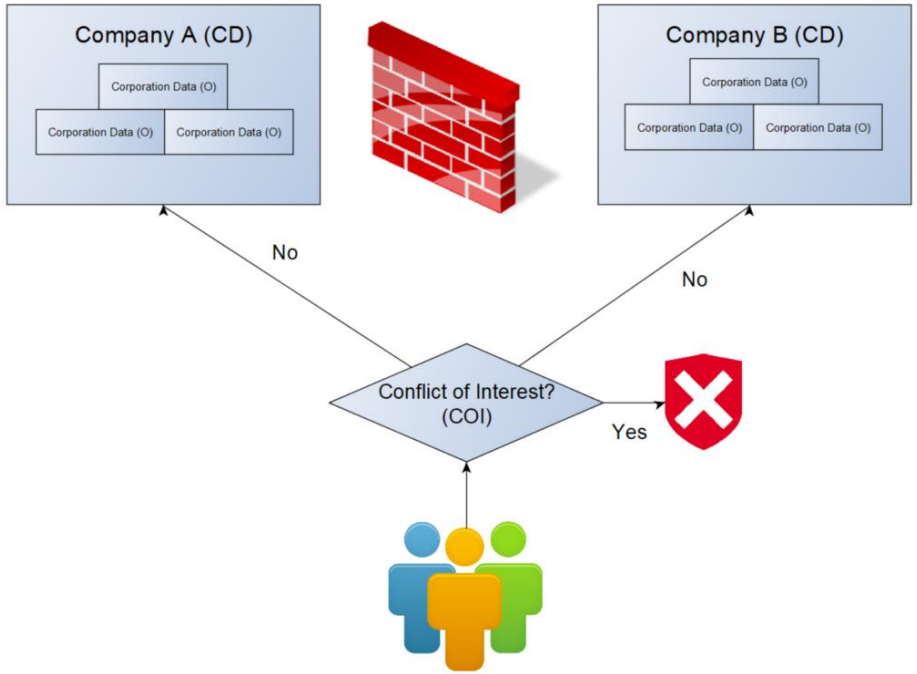
3. Clarke Wilson Model

- **Clarke Wilson Security Model** is a highly secured model where the accessibility scope of each item within a target system is carefully evaluated.
- There are ***three entities*** in this model: ***Subjects***, ***Constrained Data Items***, and ***Unconstrained Data Items***.
- There are ***two components***: ***Transformation Processes***, and ***Integration Verification Processes***.



4. Brewer-Nash Model

- **Brewer-Nash Security Model**, or Chinese Wall Model defines how to control read and write access based on the "***conflict of interest***" rules.
- Once a subject accesses a data object, he/she are automatically assigned into a group.
- From then on, the subject can only access objects of the same group.



End of the Lecture

Please don’t hesitate to raise your hand and ask questions if you're curious about anything!

Assignment 1: Threat Analysis on a Target System

Assignment Objectives:

1) This assignment aims to evaluate your skills to understand and apply threat modeling techniques in a real-world situation.
2) This assignment aims to validate that you understand syntaxes and semantics of threat modeling techniques and can use them correctly and properly.

Assignment Instruction: This assignment is counted towards 5% of your overall grade.

1) [1 Point] Students must select an information system that has full functionalities to process personal information. **Write a section to describe the target system by answering the following questions**: *What is the main purposes of this system? How many entities or components involved in the operation of this system? What are information objects flew or used in the target system? How sensitive are these information objects?*
2) [2 Points] Students must select one of the threat modeling techniques discussed in this course to analyze the target system. **Write a section to describe the rationales behind the selection of the threat modeling technique selected**. It should cover: *Why do you think this technique is the most suitable? Why are other techniques not suitable? Can this technique identify all hidden security threats?*
3) [2 Points] Students must conduct an analysis on the target system by using the selected technique. **Write a section to describe the following topics**: *Step-by-step procedure you used and what are intermediate results, the final analysis outcomes* (i.e., how many security threats are potential to be exploited?), *the criticisms of the selected technique after use* (i.e. Is it satisfying the requirements for threat modeling? Is it still the most suitable?).