Secure Software Design and Engineering
(CY-321)

Secure Design Processes

Dr. Zubair Ahmad

The Need for Secure Design

Business logic flaws addressed

Quality, maintainable software that is less prone to errors

Resilient and recoverable software

Minimal redesign and consistency:

Design Processes

When you are designing software with security in mind, certain security processes need to be established and completed

These processes are to be conducted during the initial stages of the software development project

- Attack surface evaluation
- Threat modeling
- control identification and prioritization
- documentation

Attack Surface Evaluation

A software or application's attack surface is the measure of its exposure of being exploited by a threat agent

Weaknesses in its entry and exit points that a malicious attacker can exploit to his or her advantage

- ***Targets and Enablers*** are resources that an attacker can leverage to construct an attack against the software.

- ***Channels and Protocols*** are mechanisms that allow for communication between two parties

Threat Modeling?

Performed to identify security objectives of the software, threats to the software, vulnerabilities in the software being developed

Helps the team to make design and engineering tradeoff decisions by providing insight into the areas where attention is to be prioritized and focused, from a security viewpoint

Threat modeling is a **subset of risk management** that deals specifically with **technical security risks** in software design

What Can We Threat Model?

Threat modeling is to be performed selectively, based on the value of the software as an asset to the company

It is particularly important to threat model legacy software

When there is a need to threat model legacy software, it is recommended to do so when the next version of the legacy software is being designed

We can also threat model interfaces (application programming interfaces, web services, etc.) and third-party components.

Threat modeling third-party software is often a behavioral Testing

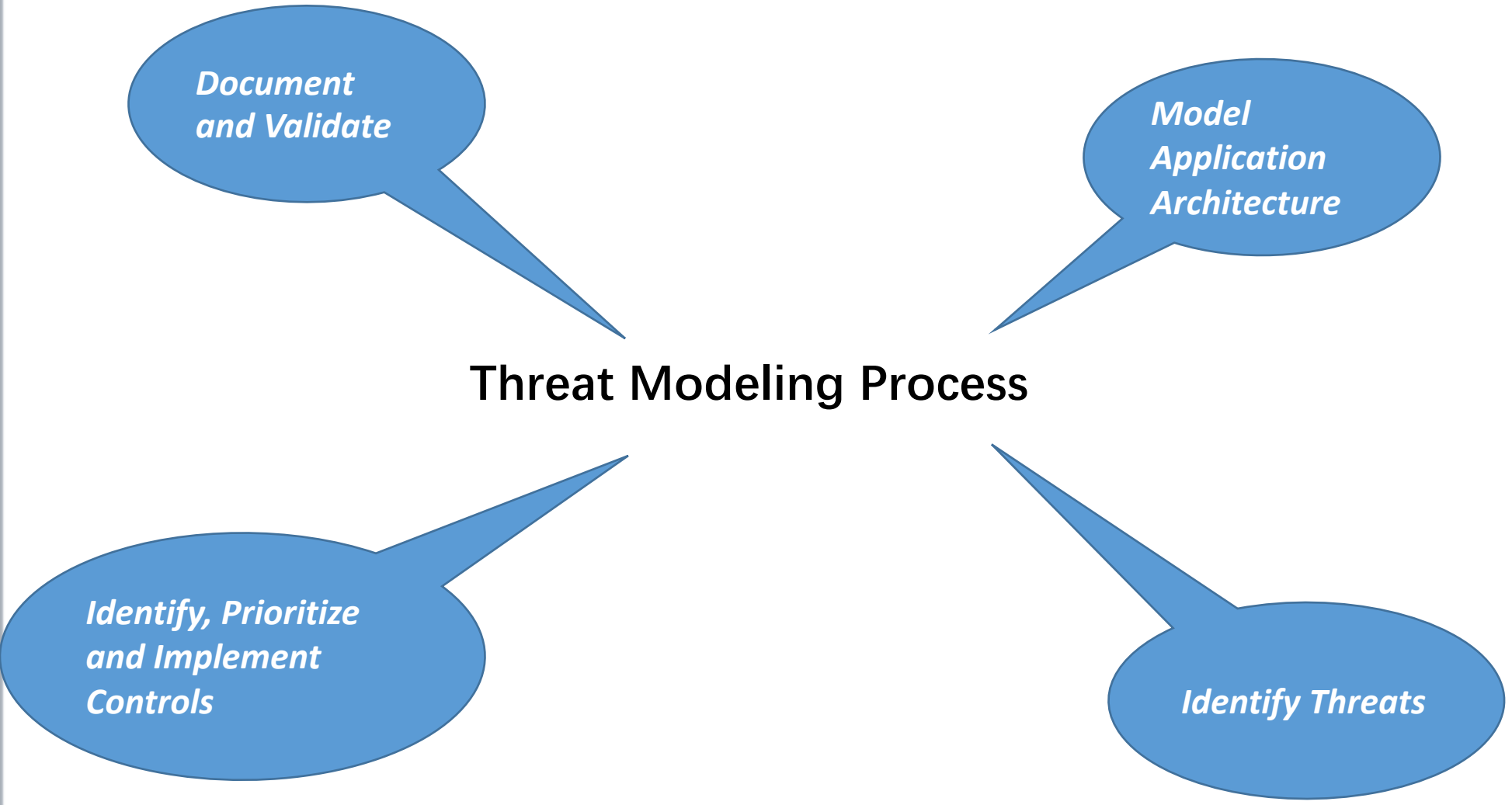
“Security Vision” for the software in threat modeling

Prevention of data theft

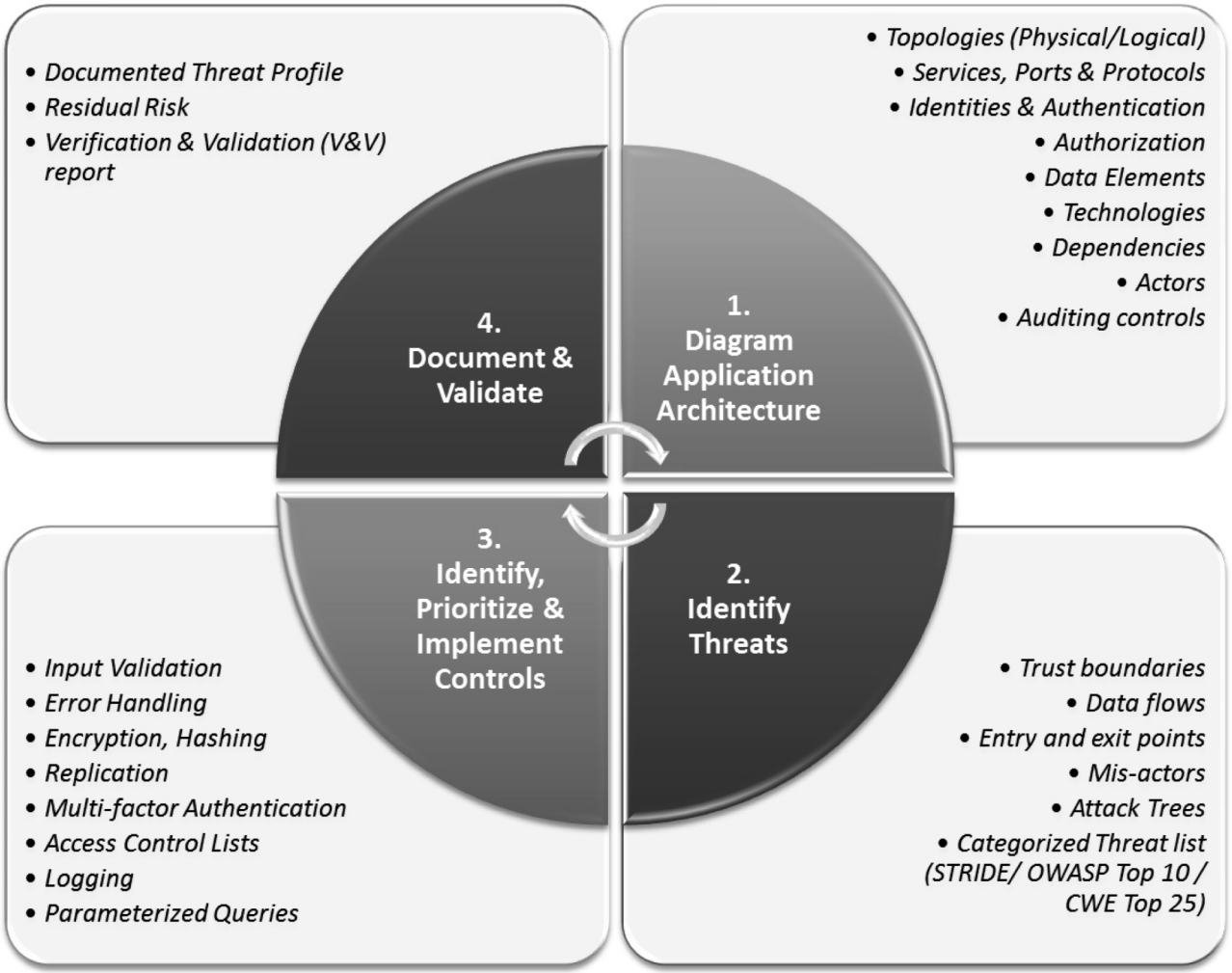
Protection of intellectual property (IP)

Provide high availability

“Security Vision” for the software in threat modeling



Threat Modeling Process



Threat Modeling Process

Model Application Architecture

Identify the software system, its components, and its interactions

Create **Data Flow Diagrams (DFDs)** or **Architecture Diagrams** to visualize data movement and trust boundaries.

Define **assets** that need protection (e.g., sensitive data, credentials, APIs).

Threat Modeling Process

Model Application Architecture

Identify the Physical Topology	Gives insight into where and how the applications will be deployed
Identify the Logical Topology	Examples include forms based, certificate based, token based, biometrics, single sign-on, multi-factor,
Identify Human and Non-Human Actors of the System	Examples include customers, sales agents, system administrators, database administrators, etc.
Identify Data Elements	Examples include product information, customer information, etc.

Threat Modeling Process

Model Application Architecture

Generate a Data Access Control Matrix.

The rights and privileges that the actors will have on the identified data elements

•Identify the technologies that will be used in building the application.

•Identify external dependencies.

Threat Modeling Process

Identify Threats

This step involves **thinking like an attacker** and listing potential security threats based on the system design.

Identify Trust Boundaries

Identify actions or behavior of the software that is allowed or not allowed

Identify Entry Points

Each entry point can be a potential threat source

Identify Exit Points

Exit points can be the source of information leakage and need to be equally protected

Threat Modeling Process

Identify Threats

This step involves **thinking like an attacker** and listing potential security threats based on the system design.

Identify Data Flows

Data flow diagrams (DFDs) and sequence diagrams assist in the understanding of how the application will accept, process, and handle data

Identify Privileged Functionality

All administrator functions and critical business transactions are identified.

Introduce Mis-Actors

Introduction of mis-actors; both human and non-human mis-actors

Threat Modeling Process

Identify Threats

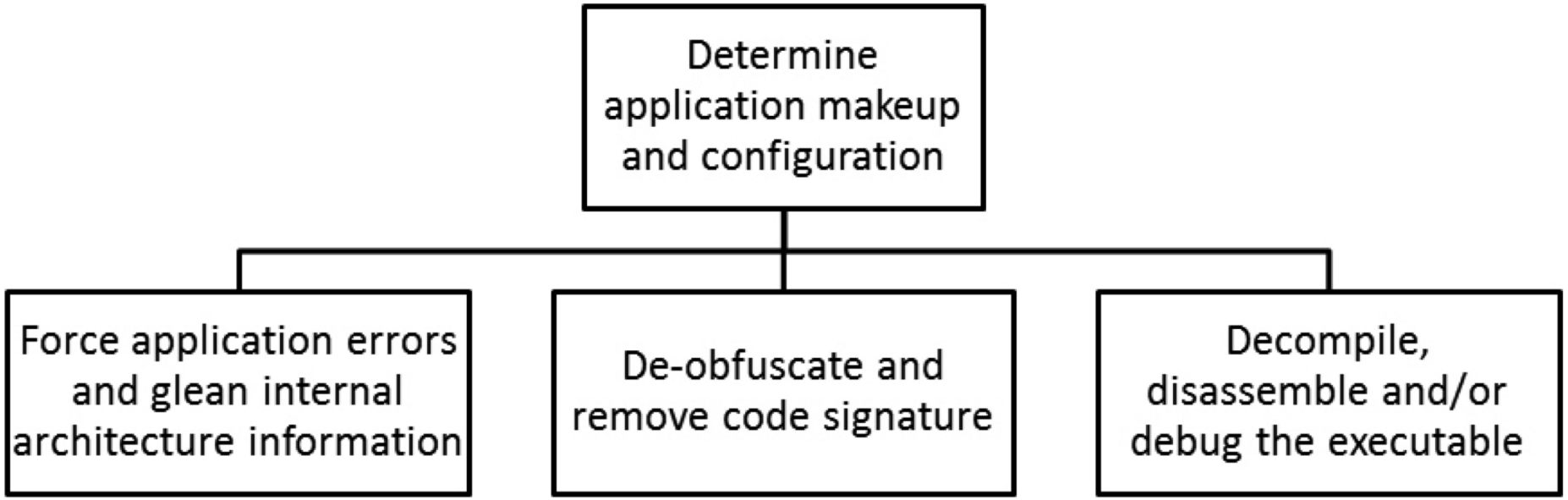
This step involves **thinking like an attacker** and listing potential security threats based on the system design.

Determine Potential and Applicable Threats

To identify relevant threats that can compromise the assets

Threat Modeling Process

Breakdown of the Attack Tree



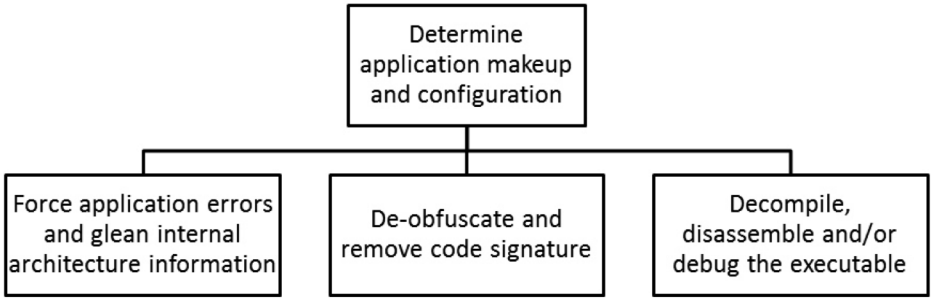
Threat Modeling Process

Breakdown of the Attack Tree

Root Node (Main Attack Goal)

"Determine application makeup and configuration"

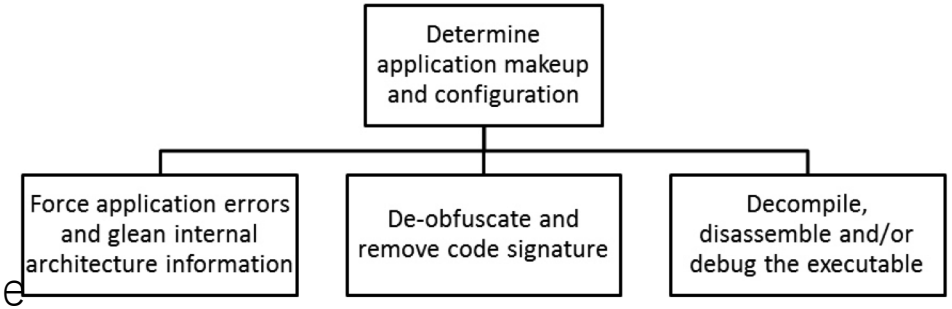
the **primary objective** of the attacker, which means they aim to gather information about how the application is built, its internal structure, and its security mechanisms



Threat Modeling Process

Breakdown of the Attack Tree

Child Nodes (Attack Techniques)



Force application errors and glean internal architecture information

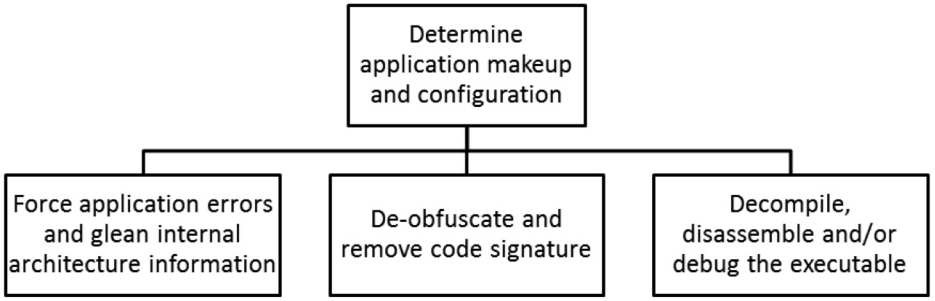
The attacker deliberately **triggers errors** in the application (e.g., malformed inputs, or unexpected requests).

Errors and debugging messages may reveal sensitive **architecture details**, API endpoints, database structures, or dependencies

Threat Modeling Process

Breakdown of the Attack Tree

Child Nodes (Attack Techniques)



De-obfuscate and remove code signature

If the application uses **obfuscation** to hide its logic, the attacker attempts to reverse-engineer it.

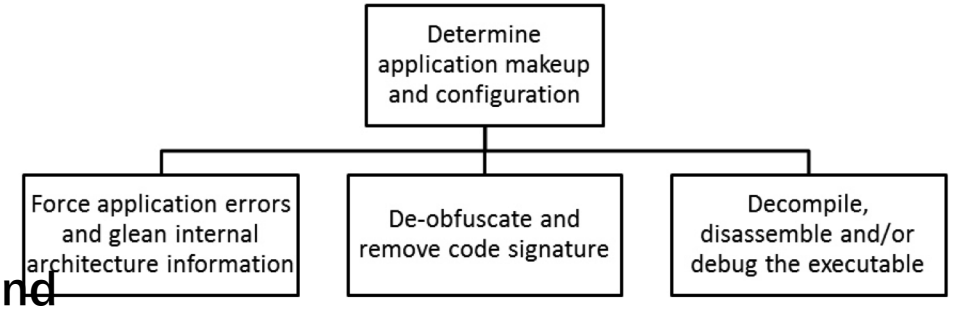
Removing a **code signature** (if present) allows attackers to modify the software without detection.

Threat Modeling Process

Breakdown of the Attack Tree

Child Nodes (Attack Techniques)

Decompile, disassemble, and/or debug the executable

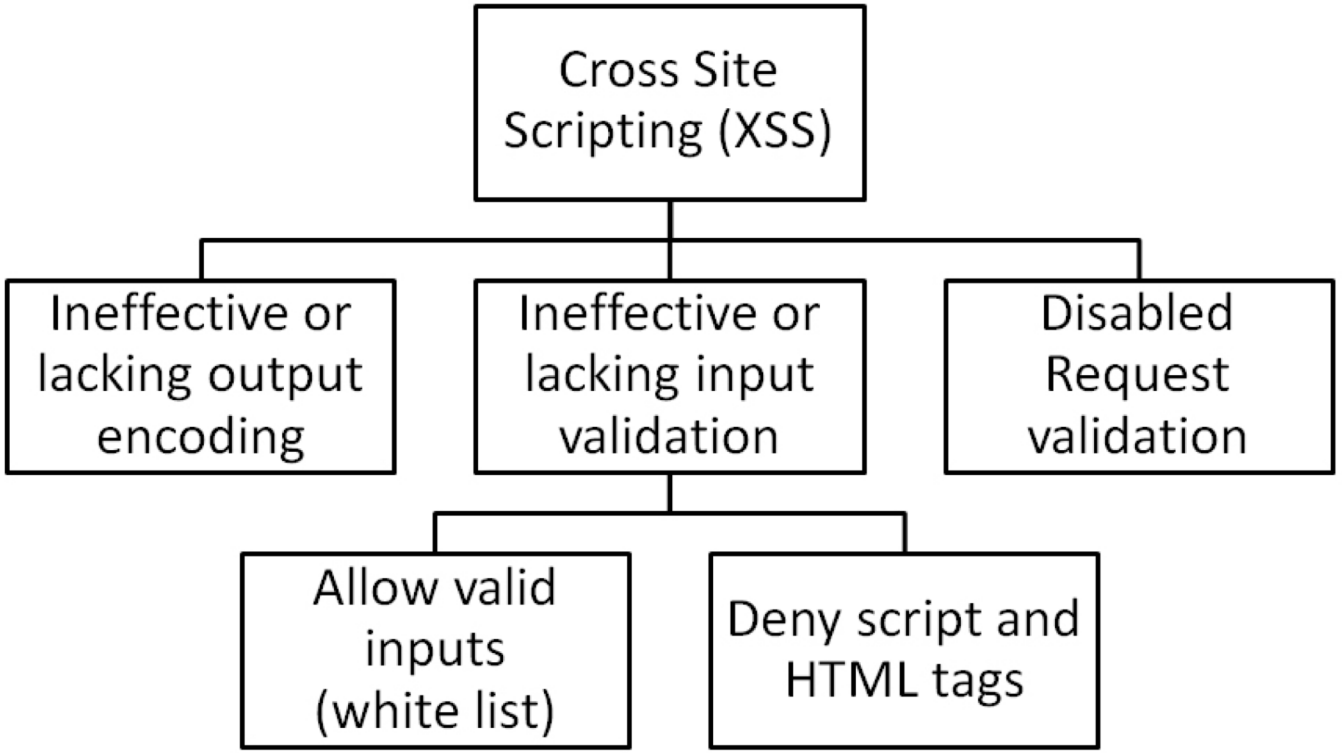


Attackers use **decompilers, disassemblers, and debuggers** to extract **source code or assembly instructions**.

This allows them to understand how the application works, find vulnerabilities, and create exploits.

Threat Modeling Process

Breakdown of the Attack Tree (Example)



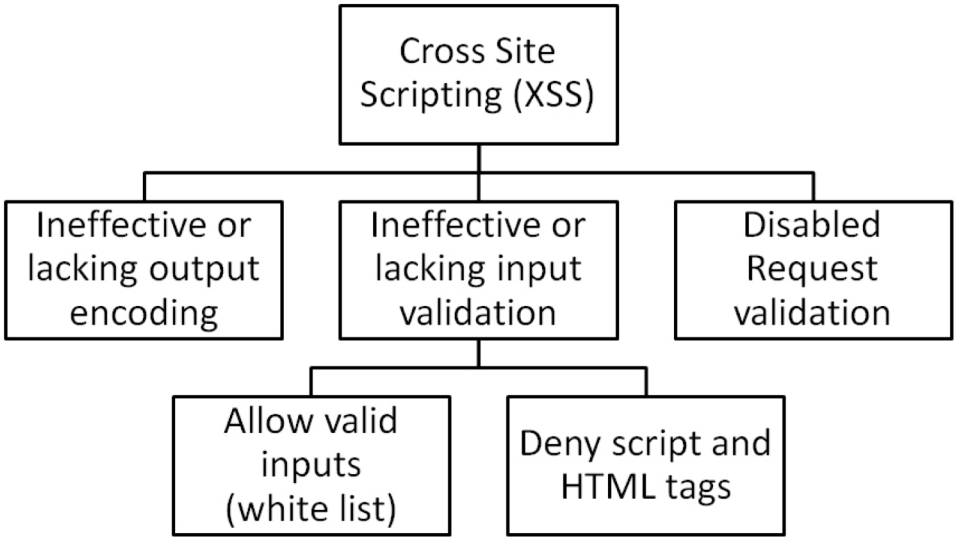
Threat Modeling Process

Breakdown of the Attack Tree (Example)

Root Node (Main Attack Goal)

Cross-Site Scripting (XSS)

The attacker aims to exploit XSS vulnerabilities to execute malicious scripts in a victim's browser



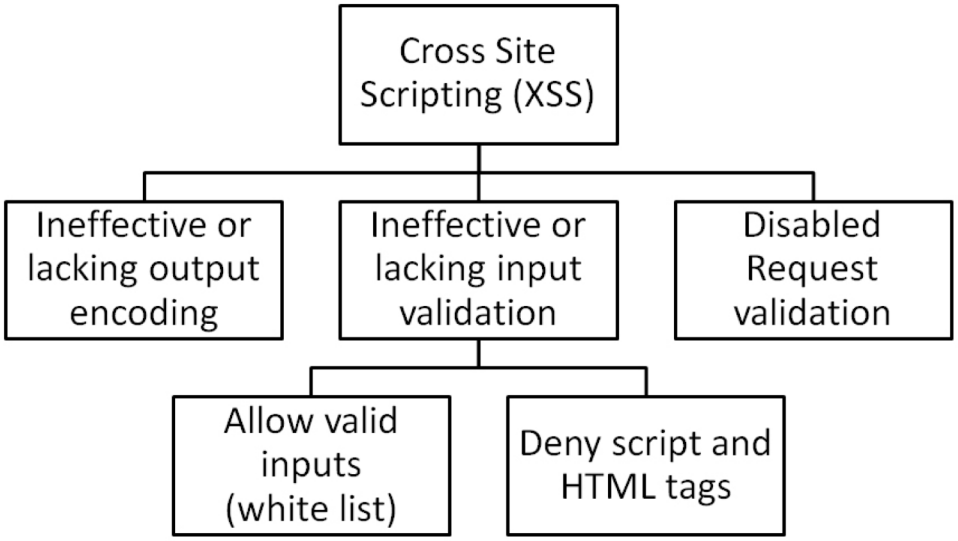
Threat Modeling Process

Breakdown of the Attack Tree (Example)

Child Nodes (Causes of XSS Vulnerabilities)

Ineffective or lacking output encoding

If the application **fails to properly encode** user input before displaying it on a webpage, attackers can inject malicious JavaScript

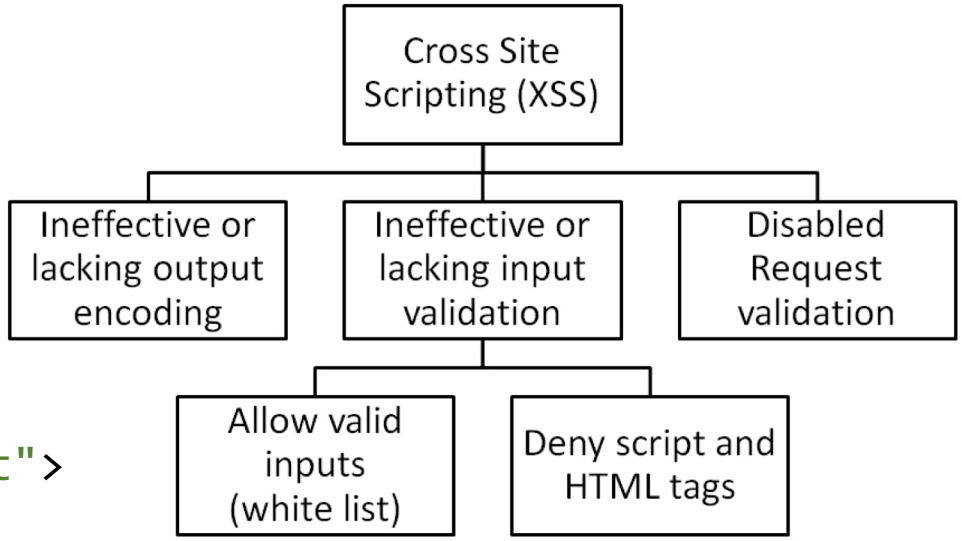


Threat Modeling Process

Breakdown of the Attack Tree (Example)

Child Nodes (Causes of XSS Vulnerabilities)

Ineffective or lacking output encoding




```
<form action="submit.php"
method="post">
    <input type="text" name="comment">
    <input type="submit">
</form>
```

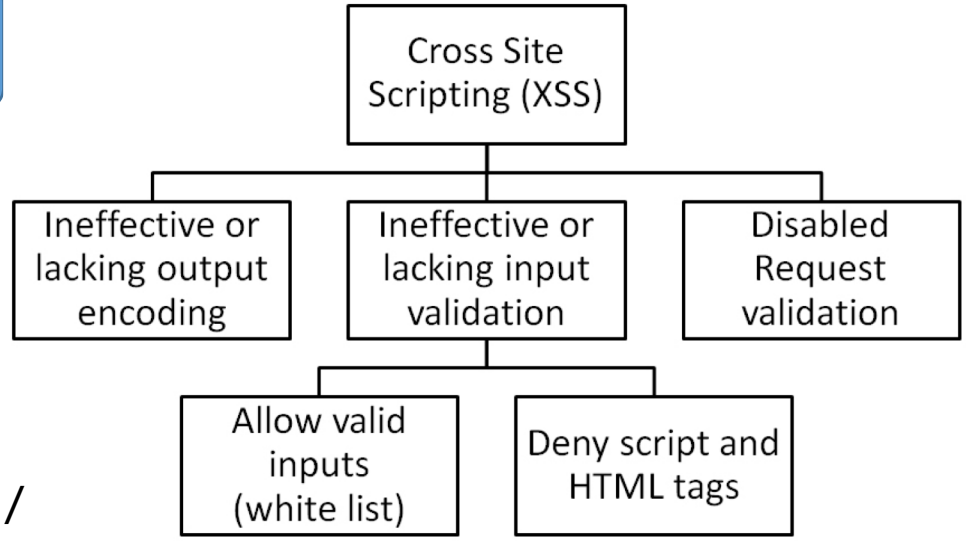
```
<script>alert('XSS')</script>
```

Threat Modeling Process

Breakdown of the Attack Tree (Example)

Child Nodes (Causes of XSS Vulnerabilities)

Ineffective or lacking output encoding



<!-- **Secure: Encode special characters** -->

```
<input type="text"
value="&lt;script&gt;alert('XSS')&lt;/
script&gt;">
```

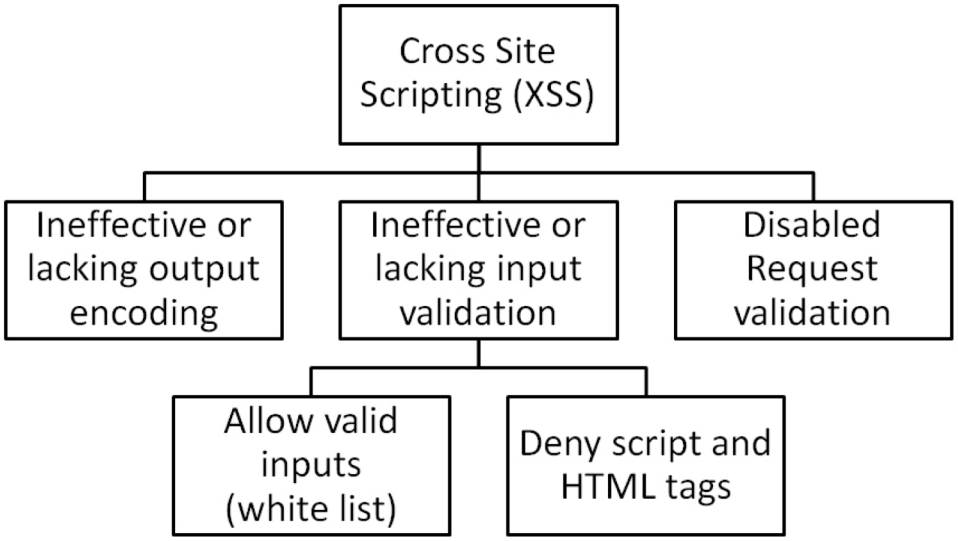
Or, using **JavaScript escaping**

```
document.write(escape("<script>alert('
XSS')</script>"));
```

Threat Modeling Process

Breakdown of the Attack Tree (Example)

Child Nodes (Causes of XSS Vulnerabilities)



Ineffective or lacking input validation

The application does not properly **validate and sanitize** user input, allowing attackers to inject scripts.

Allow valid inputs (whitelist approach)

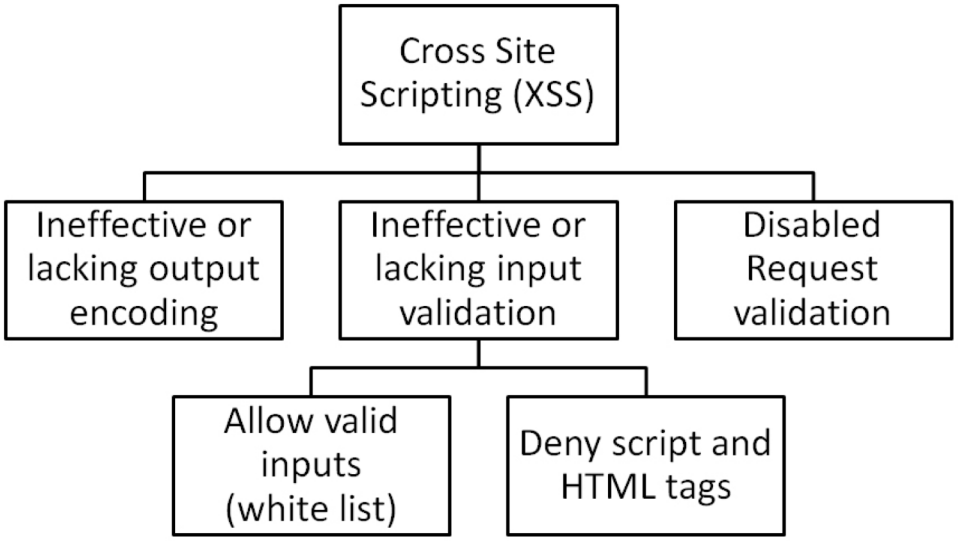
Accept only a defined set of characters (e.g., alphanumeric)

Fix: Use strict input validation (e.g., **regular expressions**, **HTML input type restrictions**).

Threat Modeling Process

Breakdown of the Attack Tree (Example)

Child Nodes (Causes of XSS Vulnerabilities)



Ineffective or lacking input validation

The application does not properly **validate and sanitize** user input, allowing attackers to inject scripts.

Deny script and HTML tags

Remove or escape **<script>, <iframe>,** and other dangerous tags

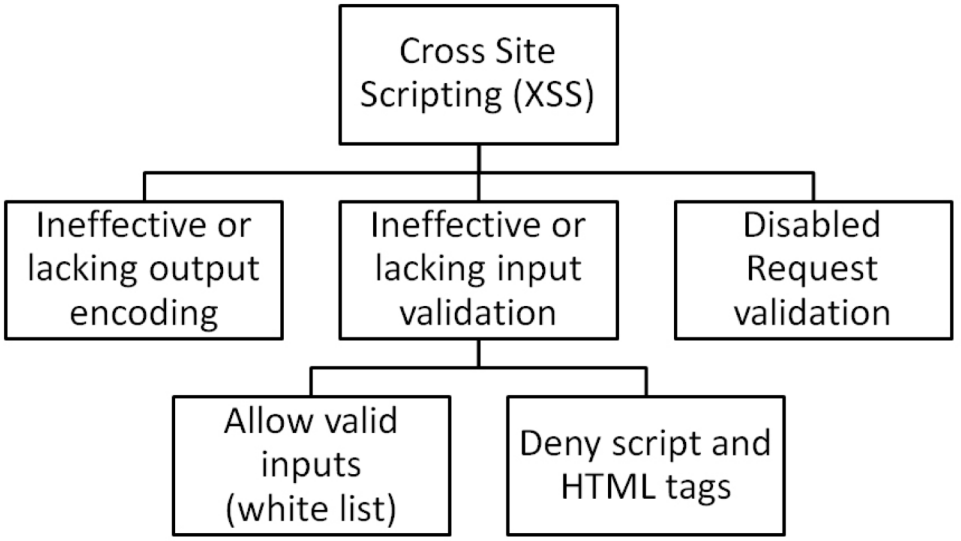
Fix: Use libraries like **DOMPurify** to sanitize HTML input

Threat Modeling Process

Breakdown of the Attack Tree (Example)

Child Nodes (Causes of XSS Vulnerabilities)

Disabled request validation



Some frameworks (like older versions of ASP.NET) provide **built-in request validation** to block suspicious inputs.

If disabled, an attacker can inject scripts via **form inputs, URLs, or cookies**.

Fix:

•Enable built-in request validation where available.
•Implement **server-side validation** instead of relying solely on client-side security.

Threat Modeling Process

Operationally Critical Threat, Asset and Vulnerability Evaluation (OCTAVE®)

Phase-1

Build asset-based threat profiles

the team identifies potential ***threats*** that can be orchestrated against each critical asset, creating a threat profile for each asset

Phase-2

Identify infrastructure vulnerabilities

Phase-1

Develop security strategy and plans

Threat Modeling Process

STRIDE

A threat modeling methodology that is performed in the design phase of software development in which threats are grouped and categorized

- ***Spoofing –*** Impersonating another user or process
- ***Tampering –*** Unauthorized alterations that impact integrity
- ***Repudiation –*** Cannot prove the action; deniability of claim
- ***Information Disclosure –*** Exposure of information to unauthorized user or process that impact confidentiality
- ***Denial of Service –*** Service interruption that impacts availability
- ***Elevation of privilege –*** Unauthorized increase of user or process

Threat Modeling Process

DREAD

Rating methodology that is often used in conjunction with STRIDE

- ***Damage potential*** – What will be the impact upon exploitability?
- ***Reproducibility*** – What is the ease of recreating the attack/exploit?
- ***Exploitability*** – What minimum skill level is necessary to launch the attack/exploit?
- ***Affected users*** – How many users will be potentially impacted upon a successful attack/exploit?
- ***Discoverability*** – What is the ease of finding the vulnerability that yields the threat?

Real-World Example: XSS Attack on Social Media Platforms

Twitter XSS Attack (2009):

- Attackers injected JavaScript into tweets.
- The script automatically reposted itself when viewed.
- Thousands of accounts were compromised.
- **Prevention**: Twitter later implemented **proper input sanitization and output encoding**.

Threat Modeling Process

Identify, Prioritize and Implement Controls

Delphi Ranking	During a Delphi risk ranking exercise, individual opinions on the level of risk for a particular threat are stated
Average Ranking	To rank the risk of the threat is to calculate the average of numeric values assigned to risk ranking categories
Probability x Impact (P x I) Ranking	The product of the probability (likelihood) of occurrence and the impact the threat will have on business operations

Threat Modeling Process

Document and Validate

The importance of documenting the threat model cannot be underestimated

Why?

because threat modeling is iterative and, through the life cycle of the project

Assignment!!

Details of Assignment will be shared today anytime before 5 pm on the course webpage. Please follow them and submit accordingly


Speaker
SESSION
acm Chapter
Abdullah Alhamdan
Ph.D Scholar at CISPA Helmholtz Center of Information Security Germany
Research Area: Software Security, Program Analysis
Title of Speech
Drilling Deep: Automatic Security Testing for JavaScript Sandboxes
12 FEBRUARY 2025 | 12:40 PM | MLH2
Organized by: Dr. Zubair

Questions??

zubair.ahmad@giki.edu.pk

Office: G14 FCSE lobby