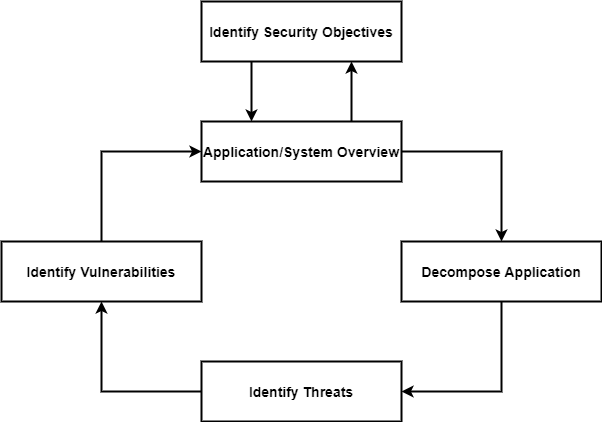
**Threat Modelling**

The threat modelling, for the purpose of penetration test or vulnerability assessment is required and it is no need to follow any traditional model schemes, however, it is demanded from the threat modelling that it should give the correct representation of the threats, their capabilities, their qualifications as per the organization being tested, and the ability to repeatedly be applied to future tests with the same results. *The focus should be on the assets and the attacker*.

It is advised whenever performing threat modelling on the attacker side, always check for the business SWOT analysis, their capabilities, and aspects of the motivation. The threat modelling is a critical step for both the tester and the organization. It provides clarity to tester mainly to know the tools, techniques, capabilities, accessibility, and general profile of the attacker and while keeping in mind what are the actual targets inside the organization. The threat modelling is about reviewing the security of any web or network infrastructure-based system. It is required to identify problem areas and determining the risk associated to each area.



The overall threat modelling process is as:

1. ***Identify Assets.***

* Overall goal of the organization.
* **Entry Points:** The identification of assets involves the entry and exit points in the application to know from where the data enters and exits.
* **Name:** The name should be assigned to the entry/exit point and identify the reasons.
* **Description:** Write the description that shows that what place entry/exit taking point and identify trust levels that exists that point.
* **Numerical ID:** Every entry and exit point have a numerical id for cross-referencing assigned to it with the threats and vulnerabilities.

1. ***Architecture Overview (survey of the system).***

* Determine components of the system.
* Routes through which data travels.
* Trust boundaries connections made.

**The Trust Levels**

**ID:** A creative number assigned to cross-reference with entry/exit points and assets.

**Name:** Create a name for the trust levels.

**Description:** Explain trust levels and its purpose.

*The rights to access and privileges should also be defined to access a resource or operation.*

1. ***Decompose Application.***

* Determining components having effect on security. For Example: Login module.

1. ***Identify Threats.***

* Enumerate any potential outside threats that system has. In general, focuses on those that are known.
* Look at the identified threats and analysis to check if the system is weak or not.
* It is required during the identification of threats that the entry/exit point should be analyzed to know how it could be attacked.

1. ***Document Threats.***

* The threats should be documented so further issues can be known and reproducibility factor reduces.

1. ***Rate Threats.***

* The threats should be rate based on their severity level. However, it can be through the framework of DREAD (damage, reproducibility, exploitability, affected users, discoverability) or CVSS (Common Vulnerability Scoring System).

# **Threat Intelligence**

The threat intelligence works in the following way:

1. **Types of threats.**
2. **Affected systems.**
3. **Detection mechanism.**
4. **Tools.**
5. **Process used to exploit vulnerabilities.**
6. **Motivation of attackers.**

# **STRIDE (A Threat Model)**

There are many threat models like:

* OCTAVE (Operationally Critical Threat Asset and Vulnerability Evaluation)
* PASTA (Process for Attack Simulation and Threat Analysis)
* Trike (focuses on modeling threats from a defensive perspective)
* VAST (Visual, Agile, and Simple Threat)
* STRIDE

*Here discussing about the STRIDE which is:*

***S = Spoofing***

***T = Tampering of data***

***R = Repudiation***

***I = Integrity***

***D = DoS***

***E = Elevation of Privileges***

Stride is a threat categorization framework. It points out six common types of threats and the security controls which are responsible for protecting against them. This is a goal-based approach where you consider the goals of an attacker which are as:

## Spoofing Identity

It is threat action aimed to access and use another user’s password and username.

**Security Control:** Authentication.

## Tampering with Data

A threat action aimed to change and modify data within system to achieve malicious goals and modify data transit between two computers.

**Security Control:** Integrity.

## Repudiation

Aim to do illegal operations in a system.

**Security Control:** Non-Repudiation.

## Information Disclosure

Aim to expose protected data; that data which is not allowed to access to a user.

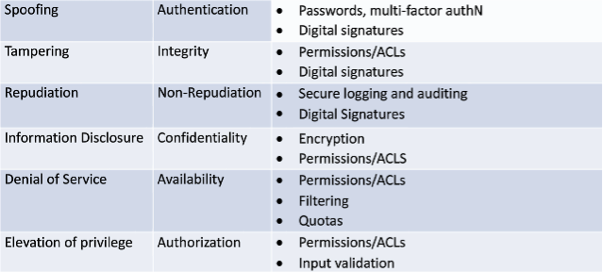
**Security Control:** Confidentiality, DoS, Availability.

## Elevation of Privileges

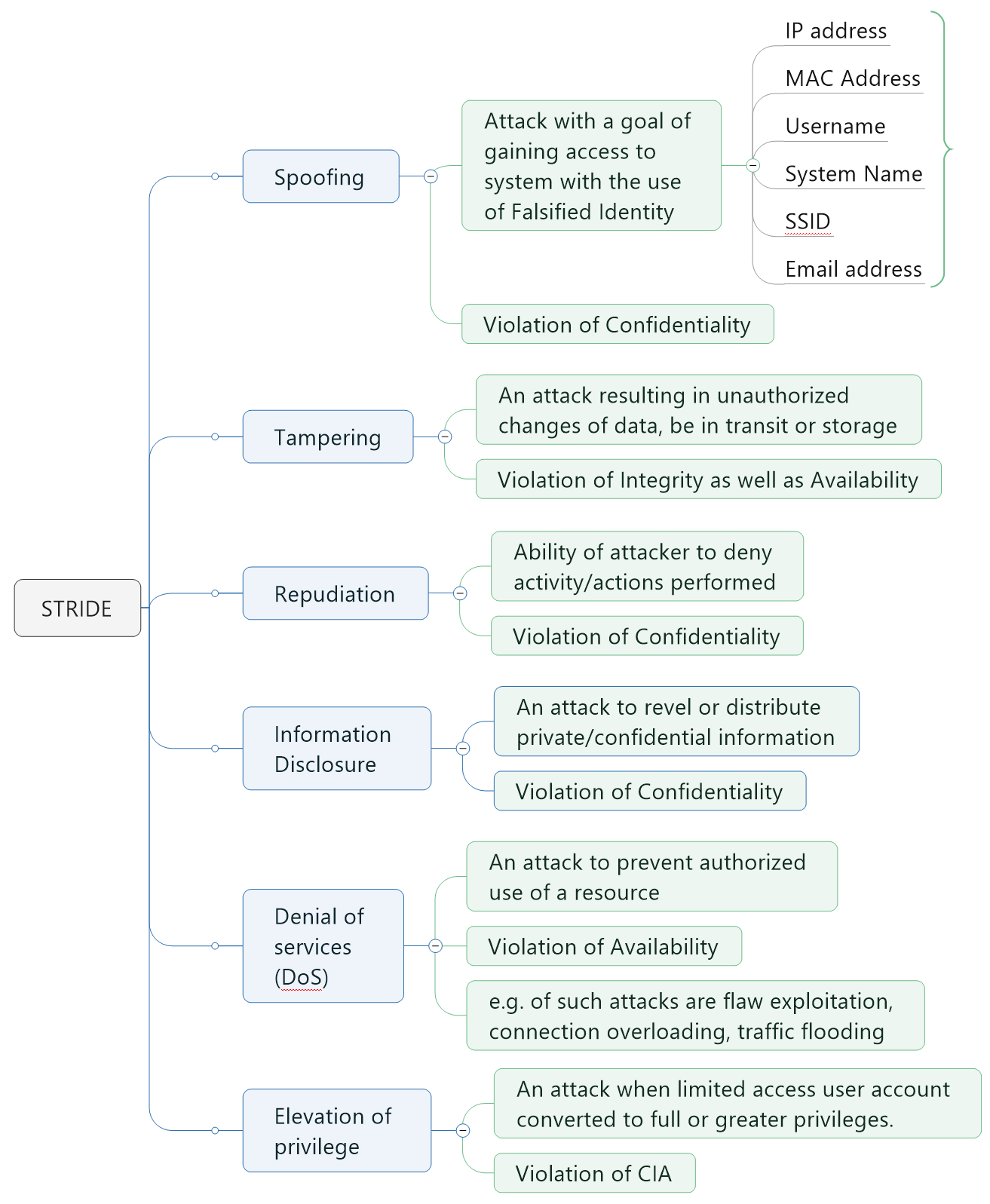
Gaining un-authorized access to resources.

**Security Control:** Authorization.

# **STRIDE SUMMARY**



# **STRIDE Tree**



# Appendix

# **Threat Agents/Community Analysis**

| **Internal** | **External** |
| --- | --- |
| Employees | Business Partners |
| Management (executive, middle) | Competitors |
| Administrators (network, system, server) | Contractors |
| Developers | Suppliers |
| Engineers | Nation States |
| Technicians | Organized Crime |
| Contractors (with their external users) | Hacktivists |
| General user community | Script Kiddies (recreational/random hacking) |
| Remote Support |  |

# **Security Control**

The primary focus of the code review is to ensure that these security controls are in place, that they work properly, and that they are correctly invoked in all the necessary places. The checklist below can help to ensure that all the likely risks have been considered.

The design and implementation approaches used for input validation, authentication, authorization, configuration management, and the remaining areas by doing this, you create a security profile for the application.

|  |  |
| --- | --- |
| **Category** | **Considerations** |
| Input validation | Is all input data validated and DV mechanism is present?  Make sure that input is not modified by a malicious user and that attacker does not inject commands or malicious data into the application. Such as HTTP headers, input fields, hidden fields, drop down lists, and other web components are validated properly.  The proper length checks on all input exist.  Make sure that the data is valid on the server side.  Can data in the database Well-formed and contained only known good chars.  A centralized model or decentralized model is used. Where in data validation.  Assure in the validation model there are no backdoors. |
| **Authentication** | Ensure that credentials secured if they are passed over the network?  Fortify strong account policies used?  Ensure that strong passwords enforced?  Ensure you are using certificates?  Assure all password verifiers used for user passwords?  Backdoors are not present in production code |
| **Authorization** | Assure that there is authorization mechanism in place and work properly.  Ensure that What gatekeepers are used at the entry points of the application?  Ensure that authorization is checked on every request.  Assure authorization fail securely and only allow access upon successful confirmation of credentials. |
| **Cookie management** | Ensure that sensitive information is not comprised.  Ensure that proper encryption is in use.  Assure the session data is being validated.  Assure that cookies contain some private information. And entire cookies are encrypted  Identify all cookies being used by the application, their name, and why they are needed. |
| **Sensitive data** | Examine What sensitive data is handled by the application?  What type of encryption is used?  Examine how are encryption keys secured? |
| **Session management** | Examine How is session generated?  Unauthenticated and authenticated.  Examine how the application tracks sessions.  Examine How is persistent session state secured as it crosses the network?  Determine the session HTTP inactivity timeout.  Determine how multithreaded/multi-user session management is performed.  Examine how the logout functionality functions |
| **Cryptography** | Examine algorithms and cryptographic techniques are used.  Does the application put its encryption into action?  How often are keys recycled?  Ensure the application is Putting known good cryptographic methods.  Ensure No important data has been transferred internally or externally |
| **Secure code**  **Environment** | Examine all memory allocations/de-allocations.  Examine the file structure.  Examine any components that should not be directly accessible available to the user?  Assure that no development environment kit is contained in the build directories. |
| **Exception management** | Determine how the application handle error conditions.  Assure that exceptions and error conditions are properly working.  Ensure resources are released if an error occurs. And no system errors can be returned to the user.  Ensure that the application fails in a secure manner. |
| **Auditing and logging** | Determine your application audit activity across all tiers on all servers?  Examine How are log files secured?  Make sure no sensitive information is logged in the event of error.  E.g., cookies, HTTP “GET” method, authentication credentials.  Make sure that successful and unsuccessful authentication is logged, and application errors are logged |