

# Utilizando Modelagem de Ameaças em uma página de Login

# Who we are...

## Tiago Zaniquelli

- Pai da Bia;
- Desde 2003 atuando com desenvolvimento;
- Desde 2021 atuando com AppSec na Conviso Application Security;
- Apaixonado por Segurança, Agilidade e Desenvolvimento;

## Redes

- Twitter: zani0x03;
- LinkedIn: <https://br.linkedin.com/in/tiago-zaniquelli>
- GitHub: <https://github.com/zani0x03>

## E-mail

- zani0x03 'at' gmail 'dot' com
- tiago.zaniquelli 'at' owasp 'dot' org

# Visão Geral

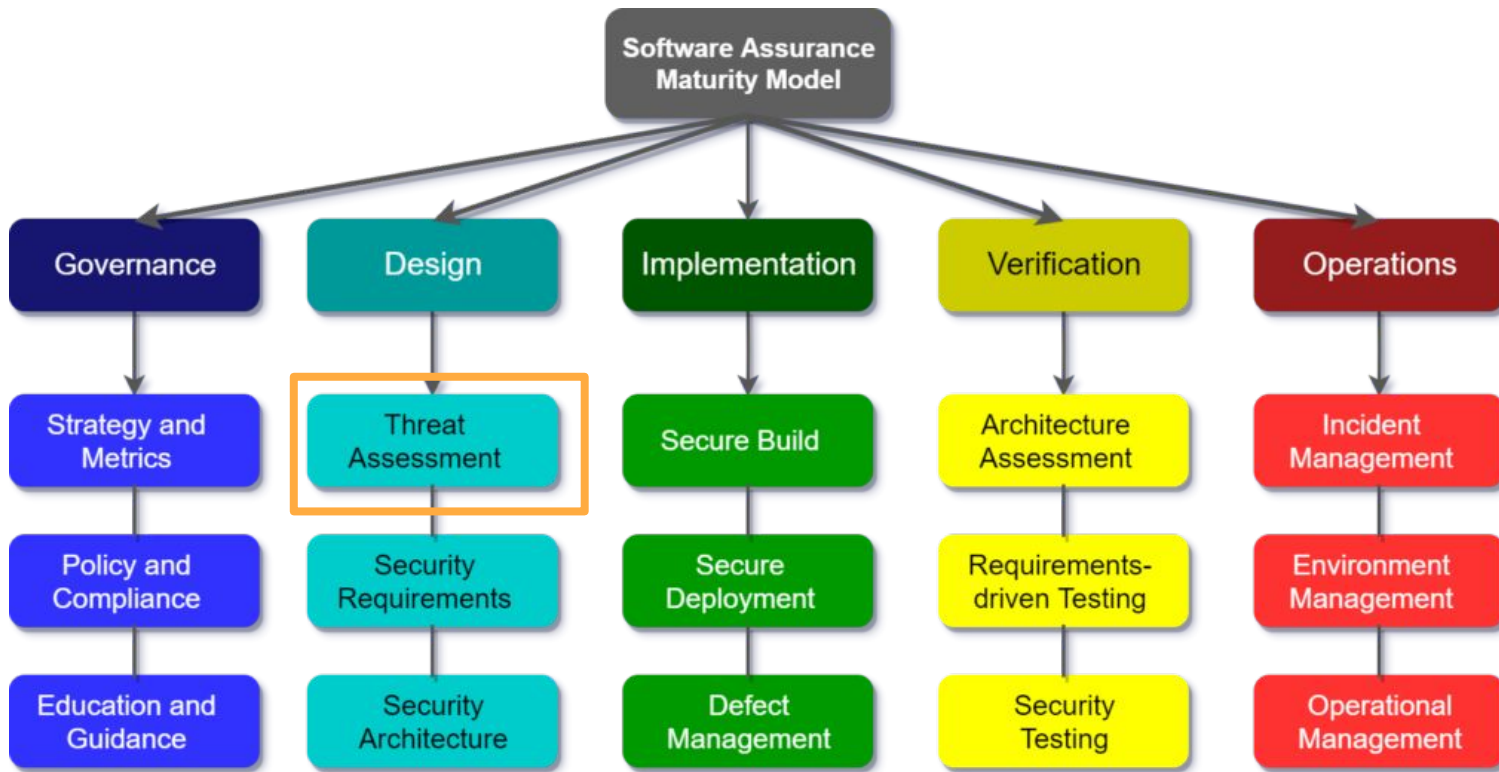
- Modelagem de ameaças;
- Modelagem de ameaças SAMM;
- CAPEC;
- CWE;
- ASVS;
- Exemplo;

# Modelagem de ameaças

# O que é modelagem de ameaças?

É uma **técnica efetiva** que ajuda a construir aplicações, sistemas, redes e serviços de **maneira segura**. De forma que **identifica ameaças potenciais** e **reduz riscos** estratégicos logo no início do ciclo de desenvolvimento.

# OWASP SAMM



# Modelagem de Ameaças dentro das práticas do OWASP SAMM Nível 1

Você identifica e gerencia falhas de projeto de arquitetura com modelagem de ameaças?

- Você **executa a modelagem de ameaças** para aplicativos de alto risco;
- Você usa listas de verificação de ameaças simples, como STRIDE;
- Você persiste o resultado de um modelo de ameaça para uso posterior.

# Modelagem de Ameaças dentro das práticas do OWASP SAMM Nível 2

Você usa uma **metodologia padrão**, alinhada aos níveis de risco de sua aplicação?

- Você treina seus arquitetos, security champions e outras partes interessadas sobre como fazer modelagem de ameaças na prática?
- Sua metodologia de modelagem de ameaças inclui pelo menos diagramação, identificação de ameaças, mitigação de falhas de design e como validar seus artefatos de modelo de ameaças?
- Mudanças no aplicativo ou contexto de negócios acionam uma revisão dos modelos de ameaças relevantes?

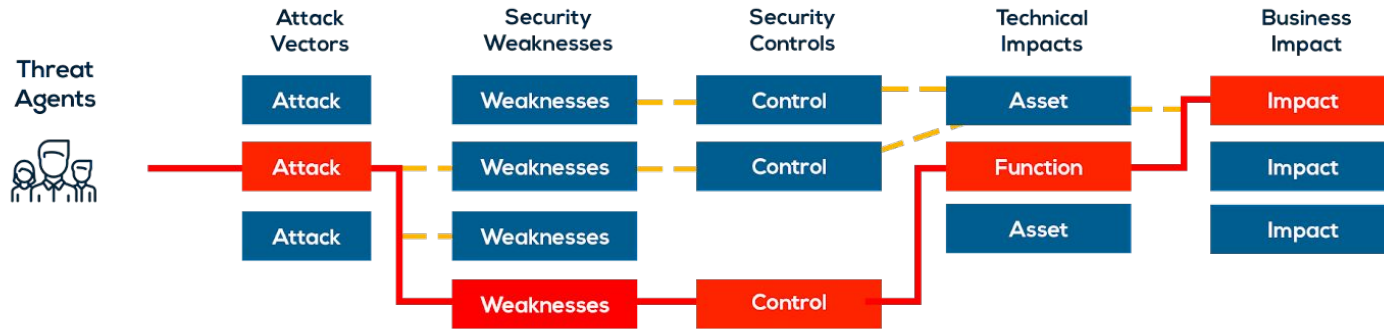


# Modelagem de Ameaças dentro das práticas do OWASP SAMM Nível 3

Você revisa e atualiza regularmente a metodologia de modelagem de ameaças para seus aplicativos?

- A metodologia do modelo de ameaça considera o feedback histórico para melhoria?
- Você regularmente (por exemplo, anualmente) revisa os modelos de ameaças existentes para verificar se nenhuma nova ameaça é relevante para seus aplicativos?
- Você automatiza partes do seu processo de modelagem de ameaças com ferramentas de modelagem de ameaças?

# Por que realizar Modelagem de ameaças?



## ○ Common Attack Pattern Enumeration and Classification

(CAPEC™) fornece um catálogo público de **padrões de ataque comuns** que ajuda os usuários a entender como os adversários exploram pontos fracos em aplicativos e outros recursos cibernéticos.

## CAPEC™ Common Attack Pattern Enumeration and Classification

A Community Resource for Identifying and Understanding Attacks

[Home](#) | [About](#) | [CAPEC List](#) | [Community](#) | [News](#) | [Search](#)

Understanding how the adversary operates is essential to effective cybersecurity. CAPEC™ helps by providing a comprehensive catalog of attack employed by adversaries to exploit known weaknesses in cyber-enabled capabilities. It can be used by analysts to advance community understanding and enhance defenses.

### CAPEC List Quick Access

#### View CAPEC

[by Mechanisms of Attack](#)

[by Domains of Attack](#)

[by Other Criteria](#)

#### Search CAPEC

ENHANCED BY Google



### New to CAPEC?

Common Attack Pattern Enumerations and Classifications (CAPEC™) can be overwhelming to someone new to cyber-attack patterns. This page offers tips on how to familiarize yourself with what CAPEC has to offer, before more fully exploring this extensive knowledge base.

### Community Engagement

[Rest API Working Group](#)

[Join the CWE/CAPEC Rest API WG](#)

# CWE™

## Common Weakness Enumeration (CWE™)

é uma lista de tipos de fraquezas de software e hardware, desenvolvida pela comunidade.

Ela serve como uma linguagem comum, uma medida para ferramentas de segurança e como uma linha de base para os esforços de **identificação, mitigação e prevenção de fraquezas.**

The screenshot shows the homepage of the Common Weakness Enumeration (CWE) website. At the top left is the logo "CWE Common Weakness Enumeration" with the tagline "A Community-Developed List of Software & Hardware Weakness Types". To the right are two circular badges: "2021 HW" and "Top 25". Below the header is a navigation bar with links for Home, About, CWE List, Scoring, Mapping Guidance, Community, News, and Search. A search bar is located on the right side of the navigation bar. The main content area features a paragraph explaining that CWE is a community-developed list of software and hardware weakness types, serving as a common language and a measuring stick for security tools. Below this is a "CWE List Quick Access" section with buttons for "View CWE" and sub-categories: "by Software Development", "by Hardware Design", "by Research Concepts", and "by Other Criteria". A "Search CWE" button is also present. To the right of the quick access section is a "2022 CWE Top 25 Most Dangerous Software Weaknesses" section, which includes a circular badge with "25" and "Top" and a description: "They are dangerous because they are often easy to find, exploit, and can allow adversaries to completely take over a system, steal data, or prevent an application from working." Below this is a link for "Software List | Limitations | Methodology | More" and a "Community Engagement" button. On the far right is a "CWE News" section with links for "News New CWE/CAPEC Board Member from University of Nebraska Omaha", "Podcast 'Using CWE/CAPEC in Education'", "News 2022 'CWE Top 25' Now Available!", and "Blog 'The Missing Piece in'".

# ASVS

O projeto OWASP Application Security Verification Standard (**ASVS**), provê uma base para testar **controles básicos** em suas aplicações web e também provê para os desenvolvedores uma **lista de requisitos de segurança** para desenvolvimento seguro.

## OWASP Application Security Verification Standard

[Main](#) | [Supporters](#) | [News and Events](#) | [Acknowledgements](#) | [Glossary](#) | [ASVS Users](#)



owasp **flagship project**

Stars ASVS

1.8k

Follow

1.6k

### What is the ASVS?

The OWASP Application Security Verification Standard (ASVS) Project provides a basis for testing web application technical security controls and also provides developers with a list of requirements for secure development.

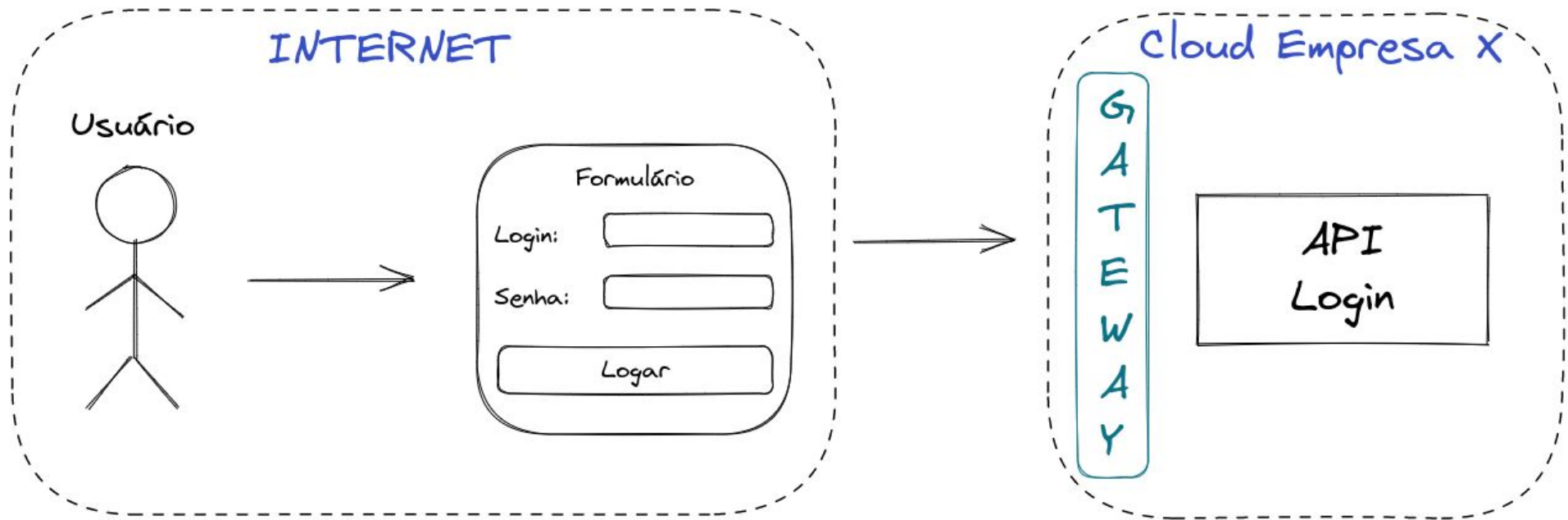
# Testes orientados a requisitos

## SAMM model overview

Governance	Design	Implementation	Verification	Operations
Strategy and Metrics	Threat Assessment	Secure Build	Architecture Assessment	Incident Management
Policy and Compliance	Security Requirements	Secure Deployment	Requirements-driven Testing	Environment Management
Education and Guidance	Security Architecture	Defect Management	Security Testing	Operational Management

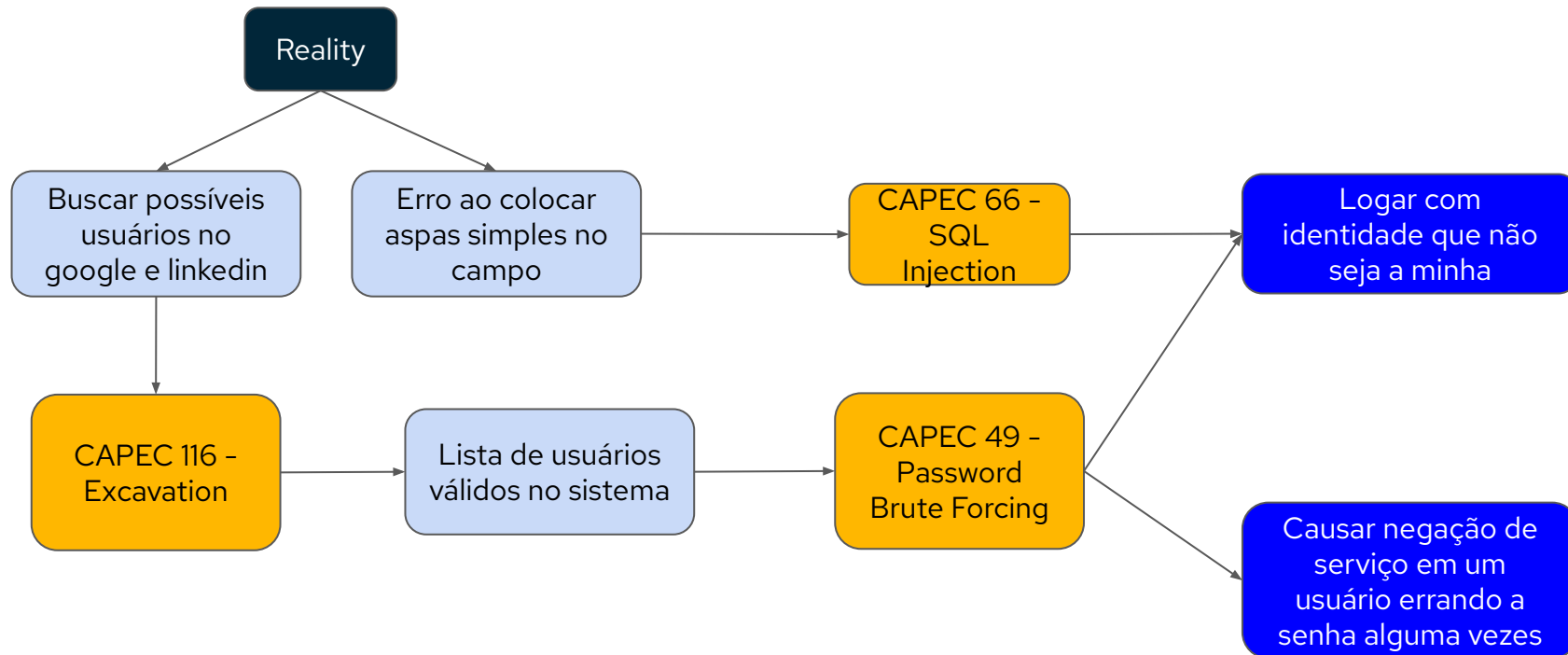
**Exemplo**

# Cenário

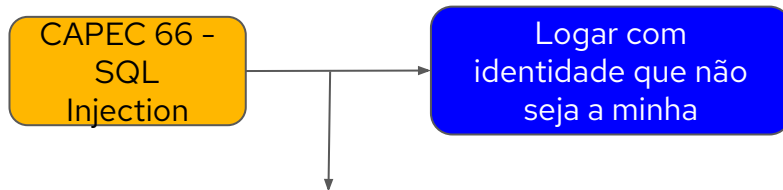




# Modelagem de Ameaças - Atacante



# Modelagem de Ameaças - Defesa



## ▼ Related Weaknesses



### CWE-ID Weakness Name

<a href="#">89</a>	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
<a href="#">1286</a>	Improper Validation of Syntactic Correctness of Input

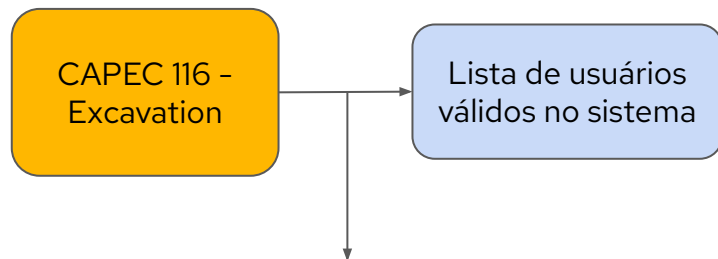
#	Description	L1	L2	L3	CWE
5.3.5	Verify that where parameterized or safer mechanisms are not present, context-specific output encoding is used to protect against injection attacks, such as the use of SQL escaping to protect against SQL injection. ( <a href="#">C3</a> , <a href="#">C4</a> )	✓	✓	✓	89

# Código Vulnerável e Código Corrigido

```
1 reference
public static async Task<User> LoginSQL(LoginRequest login)
{
    var conn = SqliteConfigConnection.GetSQLiteConnection();
    string query = "Select id, name, login, password, dateInsert, dateUpdate, isAdmin, inativo, dateChangePassword from users " +
        "where login = '"+login.Login+"' and password = '"+UtilService.ReturnSha512(login.Password)+"' and inativo = 0";
    var user = await conn.QueryAsync<User>(query);
    return user.FirstOrDefault();
}
```

```
public static async Task<User> LoginSQL(LoginRequest login)
{
    var conn = SqliteConfigConnection.GetSQLiteConnection();
    string query = "Select id, name, login, password, dateInsert, dateUpdate, isAdmin, inativo," +
        "dateChangePassword from users where login = @login and password = @password and inativo = 0";
    var user = await conn.QueryAsync<User>(query, new{
        @login = login.Login,
        @password = UtilService.ReturnSha512(login.Password)
    });
    return user.FirstOrDefault();
}
```

# Modelagem de Ameaças - Defesa



▼ Related Weaknesses

<b>CWE-ID</b>	<b>Weakness Name</b>
<a href="#">200</a>	Exposure of Sensitive Information to an Unauthorized Actor
<a href="#">1243</a>	Sensitive Non-Volatile Information Not Protected During Debug

## Authentication Responses

Using any of the authentication mechanisms (login, password reset or password recovery), an application must respond with a generic error message regardless of whether:

# Código Vulnerável e Código Corrigido

```
[HttpPost]
[Route("login")]
0 references
public async Task<ActionResult> Login([FromBody]LoginRequest login)
{
    // Recupera o usuário
    var user = await UserRepository.Login(login);

    // Verifica se o usuário existe
    if (user == null)
    {
        return Unauthorized(new
        {
            message = "User not found!"
        });
    }

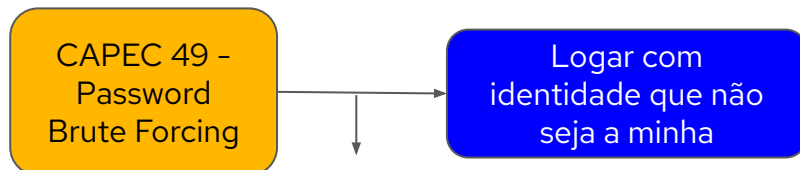
    if (user.Password == UtilService.ReturnSha512(login.Password)){
        if (login.IsAdmin.HasValue)
            user.IsAdmin = login.IsAdmin.Value;
        user.Password = "";
        var token = TokenService.GenerateToken(user);
        return Ok(new
        {
            User = user,
            token = token
        });
    }
    else{
        return Unauthorized(new
        {
            message = "Wrong password!!!"
        });
    }
}
```

```
[HttpPost]
[Route("login")]
0 references
public async Task<ActionResult> Login([FromBody]LoginRequest login)
{
    try{
        // Recupera o usuário
        var user = await UserRepository.Login(login);

        // Verifica se o usuário existe
        if (user == null)
        {
            return Unauthorized(new
            {
                message = "User/Password wrong!!!"
            });
        }

        user.Password = "";
        var token = TokenService.GenerateToken(user);
        return Ok(new
        {
            User = user,
            token = token
        });
    }catch(Exception ex){
        _logger.LogError(ex, "General error");
        return StatusCode(500, "Internal server error");
    }
}
```

# Modelagem de Ameaças - Defesa



▼ Related Weaknesses

<b>CWE-ID</b>	<b>Weakness Name</b>
<a href="#">521</a>	Weak Password Requirements
<a href="#">262</a>	Not Using Password Aging
<a href="#">263</a>	Password Aging with Long Expiration
<a href="#">257</a>	Storing Passwords in a Recoverable Format
<a href="#">654</a>	Reliance on a Single Factor in a Security Decision
<a href="#">307</a>	Improper Restriction of Excessive Authentication Attempts
<a href="#">308</a>	Use of Single-factor Authentication
<a href="#">309</a>	Use of Password System for Primary Authentication

#	Description	L1	L2	L3	CWE	<a href="#">NIST §</a>
2.2.1	Verify that anti-automation controls are effective at mitigating breached credential testing, brute force, and account lockout attacks. Such controls include blocking the most common breached passwords, soft lockouts, rate limiting, CAPTCHA, ever increasing delays between attempts, IP address restrictions, or risk-based restrictions such as location, first login on a device, recent attempts to unlock the account, or similar. Verify that no more than 100 failed attempts per hour is possible on a single account.	✓	✓	✓	<a href="#">307</a>	5.2.2 / 5.1.1.2 / 5.1.4.2 / 5.1.5.2

# Código Corrigido

```
builder.Services.Configure<IpRateLimitOptions>(options =>
{
    options.EnableEndpointRateLimiting = true;
    options.StackBlockedRequests = false;
    options.HttpStatusCode = 429;
    options.RealIpHeader = "X-Real-IP";
    options.GeneralRules = new List<RateLimitRule>
    {
        new RateLimitRule
        {
            //Endpoint = "*",
            Endpoint = "POST:/api/User/passwordrecovery",
            Period = "60s",
            Limit = 2
        },
        new RateLimitRule
        {
            Endpoint = "POST:/api/authentication/login",
            Period = "60s",
            Limit = 3
        }
    };
});
```

**Obrigado!!!**

# Referências

## **OWASP**

<https://owasp.org/www-project-top-ten/>

<https://owasp.org/www-project-application-security-verification-standard/>

<https://owaspsamm.org/>

[https://cheatsheetseries.owasp.org/cheatsheets/Authentication\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html)

## **Conviso (2022)**

<https://www.convisoappsec.com/>

## **Mitre (2022)**

<https://cwe.mitre.org/>

<https://capec.mitre.org/>

## **Common Credentials**

<https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/top-passwords-shortlist.txt>