O que o SAST não vê

O coração não sente

Um pouco de contexto...

Enquanto consultor escuto com certa frequência a palavra SAST.

Frases dos Clientes:

- Eu só preciso de uma ferramenta de SAST!
- Tenho SAST, estou seguro!
- Ao invés de um treinamento para os Devs não poderia ser uma ferramenta de SAST?
- Se eu colocar um SAST não é melhor?
- Entre outras...

SAST

O que é a Ferramenta SAST?

Uma ferramenta de SAST (Static Application Security Testing) – ou **ferramenta de análise de código** – é uma ferramenta que foi criada com o objetivo de **analisar o código fonte** ou mesmo suas versões compiladas de código, **buscando** nestes códigos **falhas** que possam comprometer a segurança.

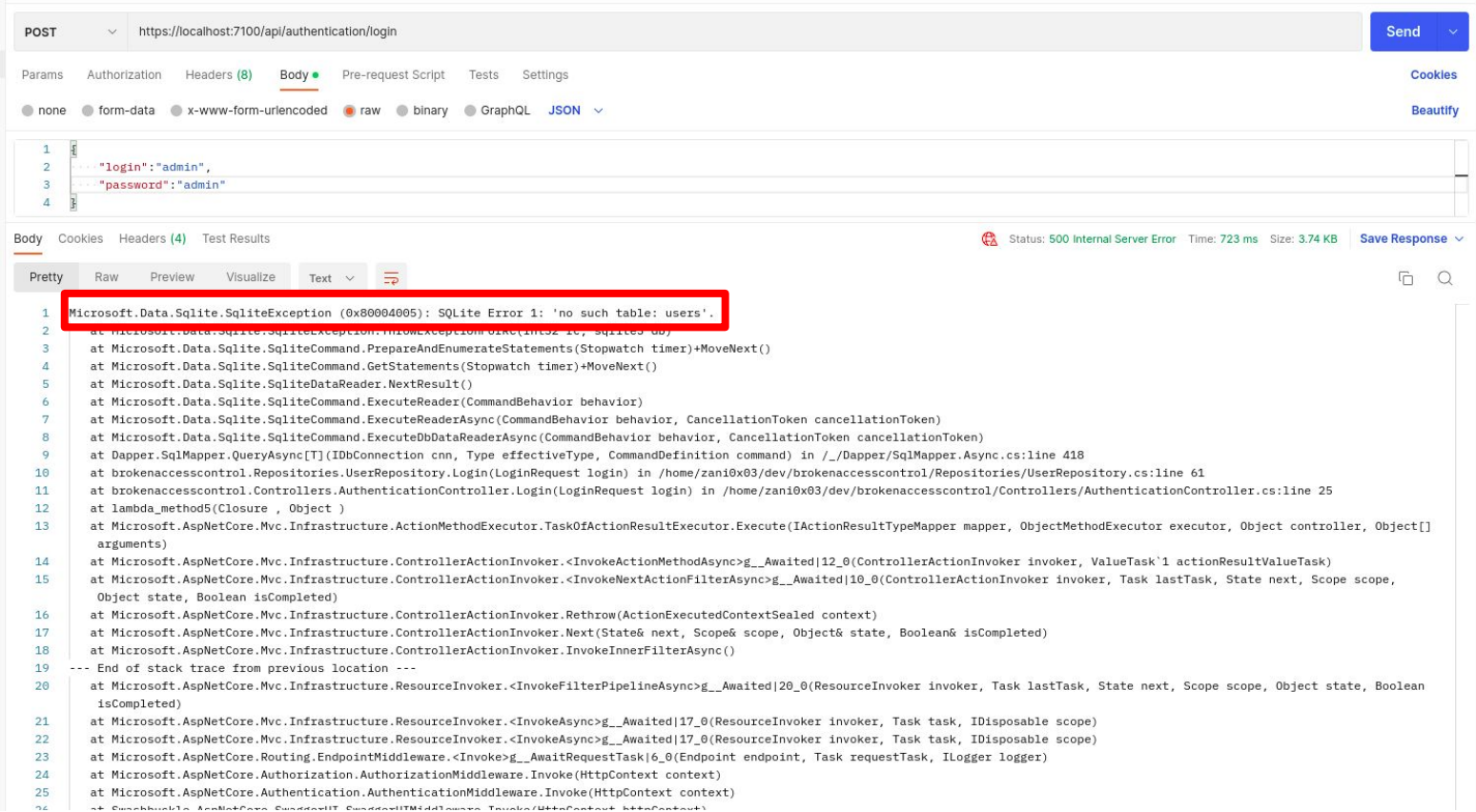
Nossa API

API desenvolvida para um workshop de vulnerabilidades.

Vulnerabilidades

- Erro não tratado;
- Enumeração de Usuários;
- Logar como admin;
- SQL Injection;
- IDOR;
- Remover item de outro usuário;
- Entre outros...

Erro não tratado



Enumeração de Usuários

Exibe uma mensagem quando se erra o usuário e outra mensagem quando erra a senha.

```
[HttpPost]
 [Route("login")]
 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!!!"
         });
     }
 }
```

Logar como admin

Ao passar um parâmetro adicional isAdmin no payload permite que um usuário se torne Admin do sistema.

```
[HttpPost]
 [Route("login")]
 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!!!"
         });
     }
 }
```

SQL Injection

Existe uma API que realiza uma consulta SQL e essa consulta está concatenada sem nenhuma tratativa da entrada de dados.

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


    // Verifica se o usuário existe
    if (user == null)
    {
        return Unauthorized(new
        {
            message = "User not found!"
        });
    }else{
        user.Password = "";
        var token = TokenService.GenerateToken(user);
        return Ok(new
        {
            User = user,
            token = token
        });
    }
}

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();
}
```

IDOR

A consulta do item cadastrado não exige autenticação, e consequentemente não verifica se quem consulta é quem cadastrou o item.

```
[HttpGet("{id}")]
public async Task<ActionResult> GetToDoItem(int id)
{
    try{
        var todoItem = await TodoItemRepository.GetToDoItem(id);

        if (todoItem != null)
            return Ok(new
            {
                todoItem
            });
        else
            return NotFound(new{
                message = "TodoItem not found!!"
            });

    }catch(Exception ex){
        _logger.LogError(ex, "General error");
        return StatusCode(500, "Internal server error");

    }
}
```

IDOR - Remover Item

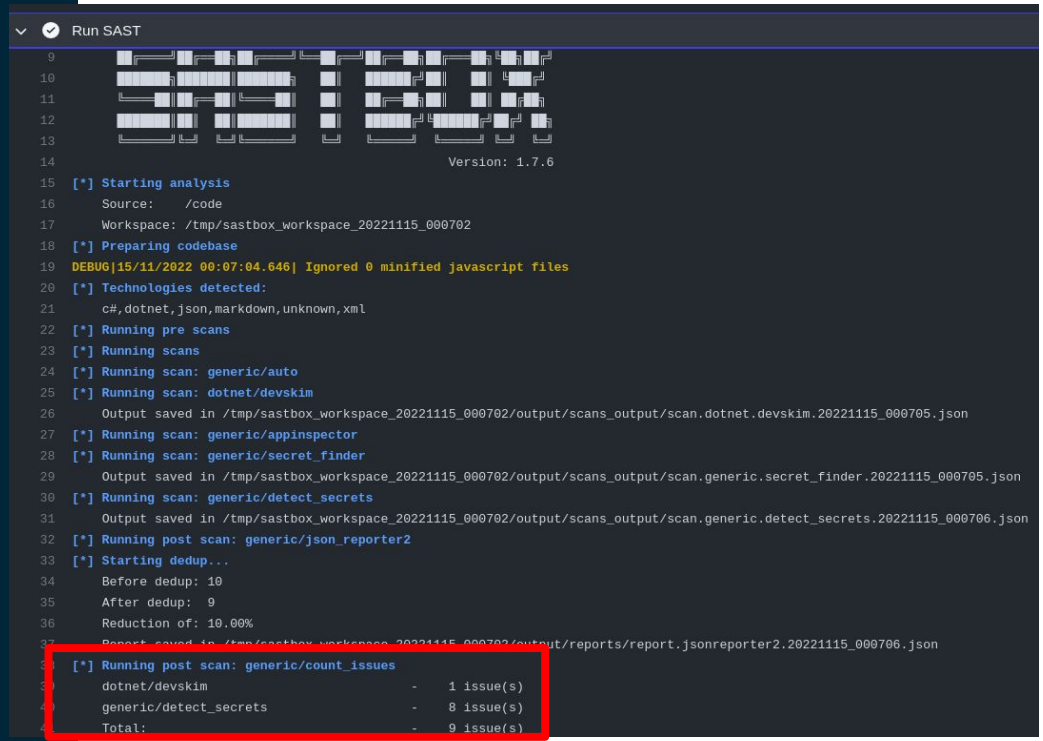
Nesse caso é solicitado que esteja logado, porém não verifica se quem está removendo é quem cadastrou o item.

```
[Authorize]
[HttpDelete("{id}")]
public async Task<ActionResult> Delete(int id)
{
    try{
        var ret = await TodoItemRepository.Delete(id);

        if (ret)
            return Ok(new
            {
                message = "Removed!"
            });
        else
            throw new Exception("Error contact the system admin!!");

    }catch(Exception ex){
        _logger.LogError(ex, "General error");
        return StatusCode(500, "Internal server error");
    }
}

public static async Task<bool> Delete(int id){

    var conn = SqliteConfigConnection.GetSQLiteConnection();
    var query = "delete from todoitems where id = @id";
    var table = await conn.ExecuteAsync(query, new{
        id = id
    });

    return table > 0 ? true : false;
}
```

Resultado das ferramentas de SAST em nossa API

1º SAST

O Resultado da 1ª ferramenta de SAST parecia animadora!



1º SAST

Só que não! Não detectou nenhuma das vulnerabilidades e ainda indicou alguns falsos positivos!

- Imagem 1 - Hex high entropy string;
- Imagem 2 - Secret keyword;
- Imagem 3 - Insecure url.

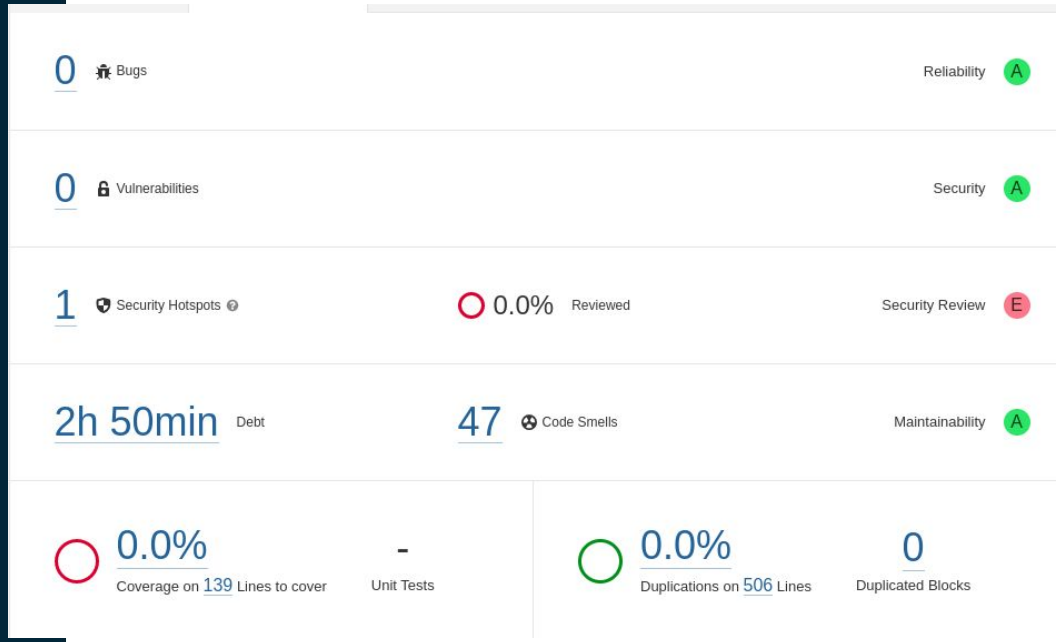
```
 7 7:       "launchBrowser": true,
 8 8:       "launchUrl": "swagger",
 9 9:       "applicationUrl": "https://localhost:7100;http://localhost:5127",
10 10:       "environmentVariables": {
11 11:         "ASPNETCORE_ENVIRONMENT": "Teste",
12 12:         "JWTSecret": "fedaf7d8863b48e197b9287d492b708e",
13 13:         "SqliteDatabase" : "./Database/brokenaccesscontrol.db"
14 14:       }
15 15:     }
16 16:   }
17 17: }
```

```
38 38:     public static async Task RecoveryPassword(PasswordRecovery recovery){
39 39:         var conn = SqliteConfigConnection.GetSQLiteConnection();
40 40:         var query = "update users set inativo=@inativo, dateChangePassword=@dateChangePassword where login=@login";
41 41:         var table = await conn.ExecuteAsync(query, new{
42 42:             inativo = 1,
43 43:             dateChangePassword = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"),
44 44:             login = recovery.Login
45 45:         });
46 46:     }
47 47:
48 48:     public static async Task<IEnumerable<User>> GetAllUsers()
49
50
```

```
1 1:
2 2: namespace brokenaccesscontrol.Models;
3 3:
4 4: public class UrlSastTest
5 5: {
6 6:     public string urlHttps = "https://www.urltestesast.com.br";
7 7:     public string urlHttp = "http://www.urltestsast.com.br";
8 8: }
9
10
```

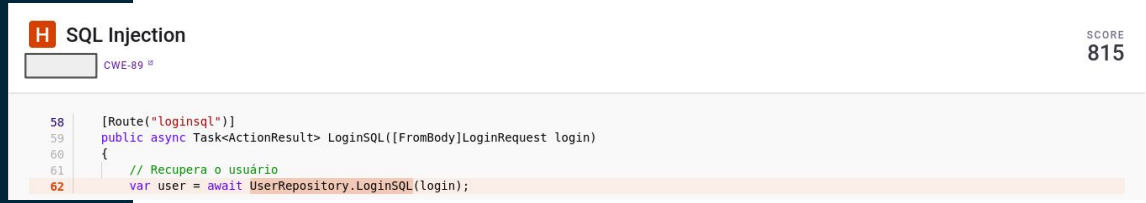
2º SAST

Resultado da 2ª ferramenta de SAST. Encontrou um problema de segurança. Falso Positivo!



3º SAST

Resultado da 3ª ferramenta de SAST. E tivemos uma melhora nos resultados. Apareceu um SQL Injection e Anti-forgery token validation disabled.

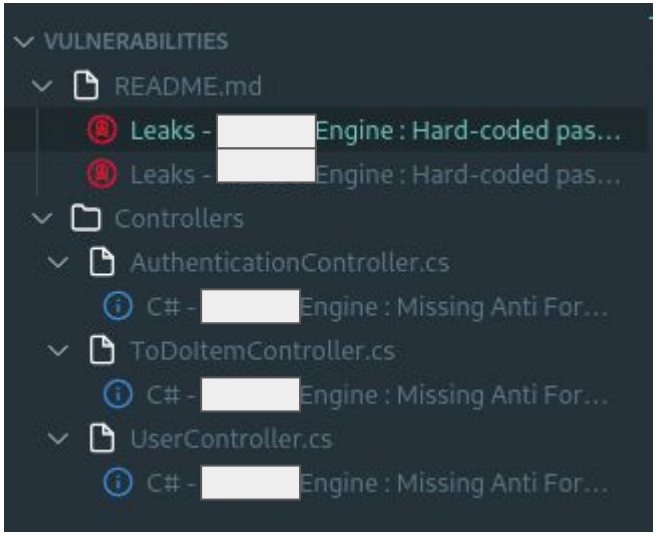


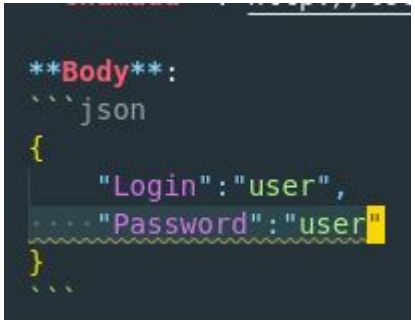
3º SAST

Um fato interessante foi que junto com o SAST executou um SCA e esse trouxe resultados interessantes!

4º SAST

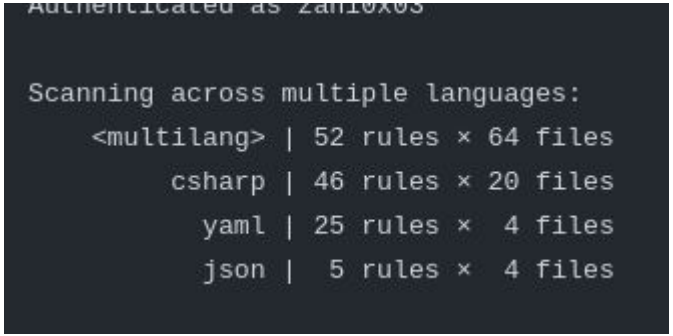
Resultado da 4ª ferramenta de SAST onde os Hard-coded password no arquivo README.md são falsos positivos.





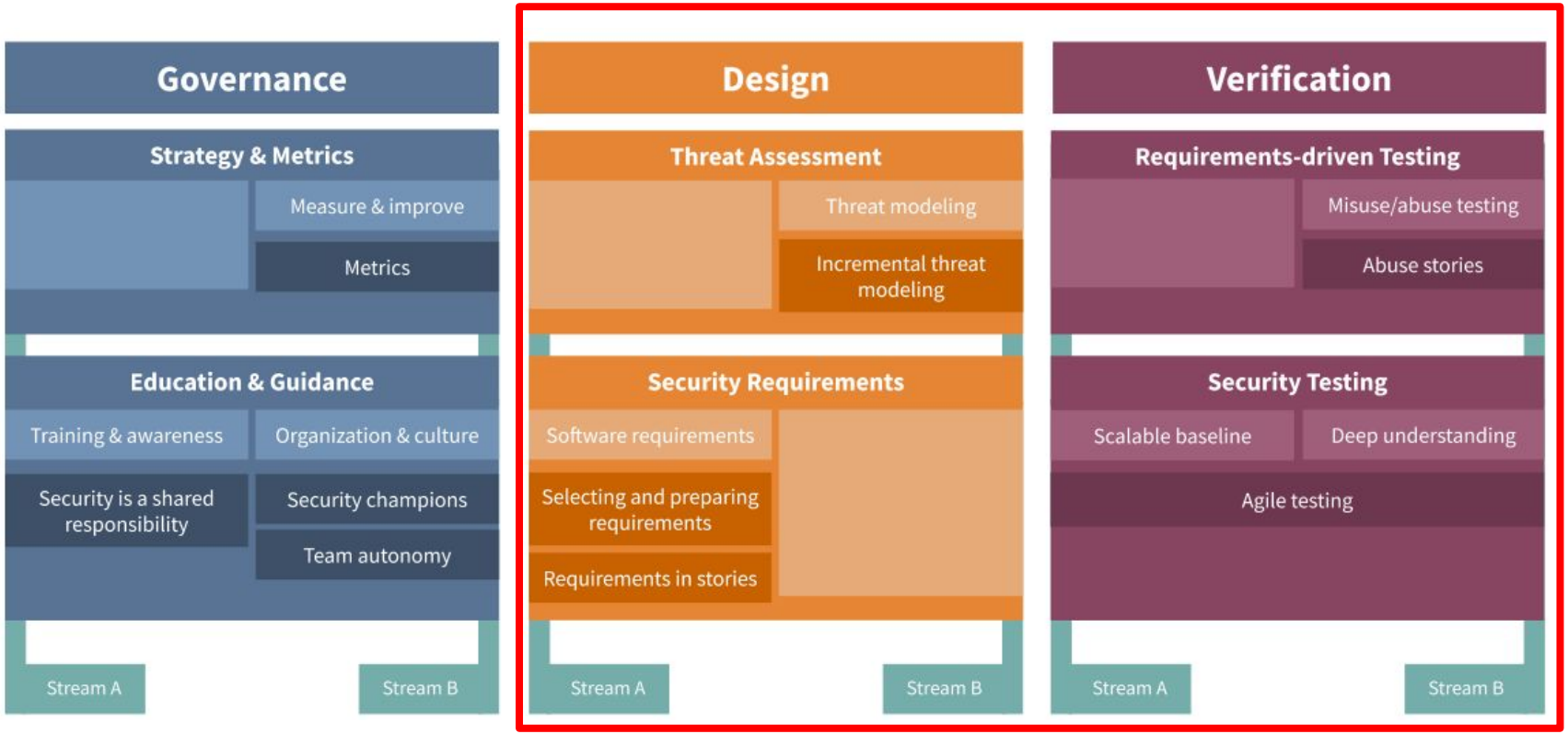
5º SAST

Não detectou nada!



E agora quem poderá nos ajudar???

EUUUU!!!!! OWASP SAMM Agile Guidance



OWASP ASVS

O que é o OWASP ASVS?

O projeto **OWASP Application Security Verification Standard** (ASVS) fornece uma base para testar os **controles técnicos de segurança de aplicativos da Web** e também fornece aos desenvolvedores **uma lista de requisitos** para um **desenvolvimento seguro**.

Requisitos de Segurança - ASVS

Capítulo	Seção	Descrição	CWE
Autenticação	Segurança de Senha	Verificar se as senhas definidas pelo usuário têm pelo menos 12 caracteres (após a combinação de vários espaços). ([C6](https://owasp.org/www-project-proactive-controls/#div-numbering))	521
Controle de acesso	Controle de acesso ao nível de operação	Verificar se os dados sensíveis e as APIs estão protegidos contra ataques Insecure Direct Object Reference (IDOR) visando a criação, leitura, atualização e exclusão de registros, como criar ou atualizar o registro de outra pessoa, visualizar os registros de todos ou excluir todos os registros.	639
Validação, higienização e codificação	Codificação de saída e prevenção de injeção	Verificar se a seleção de dados ou consultas de banco de dados (por exemplo, SQL, HQL, ORM, NoSQL) usam consultas parametrizadas, ORMs, estruturas de entidade ou estão protegidas contra ataques de injeção de banco de dados. ([C3](https://owasp.org/www-project-proactive-controls/#div-numbering))	89
Tratamento e registro de erros	Tratamento de erros	Verificar se uma mensagem genérica é exibida quando ocorre um erro inesperado ou sensível à segurança, possivelmente com um ID exclusivo que a equipe de suporte pode usar para investigar. ([C10](https://owasp.org/www-project-proactive-controls/#div-numbering))	210

Modelagem de ameaças

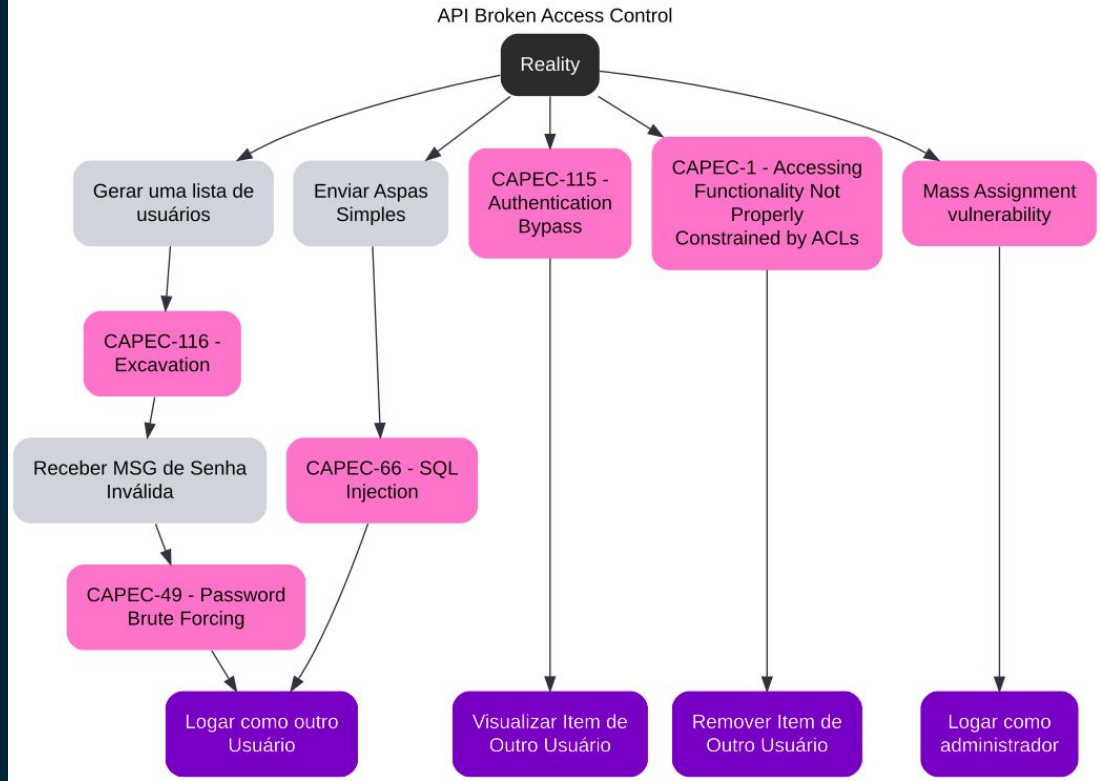
O que é modelagem de ameaças?

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.

Modelagem de Ameaças

Relação Ataque x CWE

CAPEC-116: CWE-200, CWE-1243;
CAPEC-49: CWE-521, CWE-262, CWE-263, CWE-257, CWE-654, CWE-307, CWE-308, CWE-309;
CAPEC-66: CWE-89, CWE-1286;
CAPEC-115: CWE-287;
CAPEC-1:CWE-276, CWE-285, CWE-434, CWE-693, CWE-732, CWE-1191, CWE-1193, CWE-1220, CWE-1297, CWE-1311, CWE-1314, CWE-1315, CWE-1318, CWE-1320, CWE-1321, CWE-1327.
Mass Assignment: CWE-915.

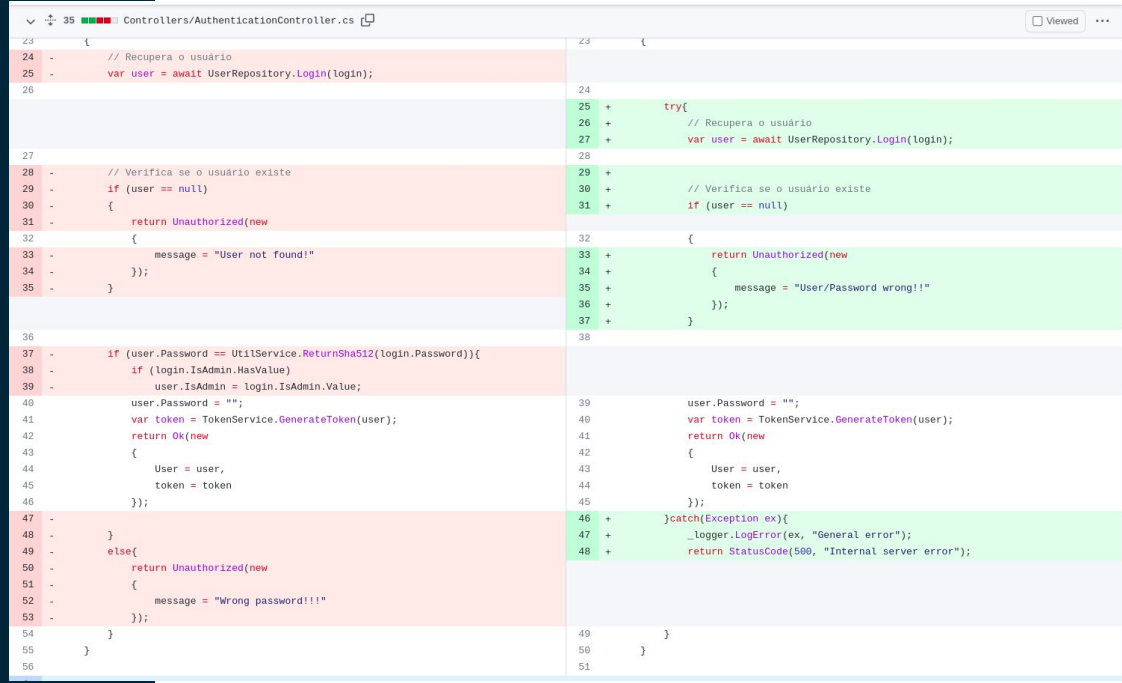


Code Review

Code Review

Code review tem por objetivo identificar falhas de segurança na aplicação relacionadas a design e features, juntamente com a causa raiz.

Como mencionado, a API foi feita para um workshop. Os desenvolvedores realizaram um code review nela e demoraram cerca de 20 minutos para encontrarem as vulnerabilidades.



35 Controllers/AuthenticationController.cs — Viewed

```
-       // Recupera o usuário
-       var user = await UserRepository.Login(login);

+       try{
+           // 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!!!"
-           });
        }
    }

+           // 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");
        }
    }
```

Aplicando os requisitos de segurança em nossa API

Erro não tratado

Aplicado o requisito de segurança do ASVS - Verificar se uma mensagem genérica é exibida quando ocorre um erro inesperado ou sensível à segurança, possivelmente com um ID exclusivo que a equipe de suporte pode usar para investigar. ([C10](https://owasp.org/www-project-proactive-controls/#div-numbering))

```
[HttpPost]
[Route("login")]
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");
    }
}
```

Enumeração de Usuários

Aplicado orientação de mensagem genérica “User/Password wrong” conforme documento Authentication Cheat Sheet.

```
[HttpPost]
[Route("login")]
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");
    }
}
```

Logar como admin

Aplicado o requisito de segurança do ASVS - Verificar se a aplicação tem limites de lógica de negócios ou validação para proteger contra prováveis riscos ou ameaças de negócios, identificados usando modelagem de ameaças ou metodologias semelhantes.

```
[HttpPost]
[Route("login")]
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");
    }
}


public static async Task<User> Login(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();
}
```

SQL Injection

Aplicado o requisito de segurança do ASVS - Verificar se a seleção de dados ou consultas de banco de dados (por exemplo, SQL, HQL, ORM, NoSQL) usam consultas parametrizadas, ORMs, estruturas de entidade ou estão protegidas contra ataques de injeção de banco de dados. ([C3](https://owasp.org/www-project-proactive-controls/#div-numbering))

```
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();
}
```

IDOR

Aplicado o requisito de segurança do ASVS - Verificar se os dados sensíveis e as APIs estão protegidos contra ataques Insecure Direct Object Reference (IDOR) visando a criação, leitura, atualização e exclusão de registros, como criar ou atualizar o registro de outra pessoa, visualizar os registros de todos ou excluir todos os registros.

```
[Authorize]
[HttpGet("{id}")]
public async Task<ActionResult> GetToDoItem(int id)
{
    try{
        var todoItem = await TodoItemRepository.GetToDoItem(id,
            ((ClaimsIdentity)User.Identity).FindFirst("UserId").Value);

        if (todoItem != null)
            return Ok(new
            {
                todoItem
            });
        else
            return NotFound(new{
                message = "TodoItem not found!!"
            });

    }catch(Exception ex){
        _logger.LogError(ex, "General error");
        return StatusCode(500, "Internal server error");
    }
}

public static async Task<TodoItem> GetToDoItem(int id, string userId)
{
    var conn = SqliteConfigConnection.GetSQLiteConnection();
    string query = "Select id, name, description, userId from todoitems "+
        "where id = @id and userId = @userId";
    var todoItem = await conn.QueryAsync<TodoItem>(query, new{
        id = id,
        userId = userId
    });
    return todoItem.FirstOrDefault();
}
```

IDOR - Remover Item

Aplicado o requisito de segurança do ASVS - Verificar se os dados sensíveis e as APIs estão protegidos contra ataques Insecure Direct Object Reference (IDOR) visando a criação, leitura, atualização e exclusão de registros, como criar ou atualizar o registro de outra pessoa, visualizar os registros de todos ou excluir todos os registros.

```
[Authorize]
[HttpDelete("{id}")]
public async Task<ActionResult> Delete(int id)
{
    try{
        var ret = await TodoItemRepository.Delete(id,
            ((ClaimsIdentity)User.Identity).FindFirst("UserId").Value);

        if (ret)
            return Ok(new
            {
                message = "Removed!"
            });
        else
            throw new Exception("Error contact the system admin!!");

    }catch(Exception ex){
        _logger.LogError(ex, "General error");
        return StatusCode(500, "Internal server error");
    }
}

public static async Task<bool> Delete(int id, string userId){

    var conn = SqliteConfigConnection.GetSQLiteConnection();
    var query = "delete from todoitems where id = @id and userId = @userId";
    var table = await conn.ExecuteAsync(query, new{
        id = id,
        userId = userId
    });

    return table > 0 ? true : false;
}
```

Let's empower developers to build secure applications?!

Referências

- OWASP Application Security Verification Standard. Disponível em: https://owasp.org/www-project-application-security-verification-standard/. Acesso em: 17 nov. 2022.
- OWASP Code Review Guide. Disponível em: https://owasp.org/www-project-code-review-guide/. Acesso em: 17 nov. 2022.
- SAMM AGILE GUIDANCE. Disponível em: https://owaspsamm.org/guidance/agile/. Acesso em: 17 nov. 2022.
- Authentication Cheat Sheet. Disponível em: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html. Acesso em: 17 nov. 2022.
- MAUÉS, Rodrigo. Code Review e SAST: entenda a diferença, 2019. Disponível em: https://blog.convisoappsec.com/diferenca-entre-code-review-e-sast/. Acesso em: 18 nov. 2022.

Obrigado!

Tiago Zaniquelli
Security Analyst

tiago.zaniquelli@owasp.org
https://zani0x03.me/
zani0x03