República De Angola

Ministério da Educação

Instituto Superior Politécnico Independente

Curso de Engenharia Informática

Segurança da Informação

Relatório de Implementações de Segurança Do Sistema HealthCenter

Membros do grupo

Grupo nº: 6

Nome	Nº de Estudante	Nota
Doroteia Tyimboto	210239	
Esperança João	210482	
Elias Soares	210874	
Emerson de Brito	210321	
Jorge Silva	211692	

Orientator: Engº Hernani Lote

_____

Introdução

A segurança nas aplicações hoje em dia é um tópico muito abrangente, sendo não só aplicável a nível das aplicações mas a nivel das pessoas a fim de se ter mais privacidade dos dados importantes e como é crescente o nível de ameças a segurança, devemo-nos prevenir de qualquer vazamento.

Implementações de Segurança

1- Autenticação Segura

a) Implementação de login seguro com armazenamento de senha utilizando hash.

O ASP.NET Identity Framework, por padrão, utiliza o algoritmo de hash PBKDF2 (Password-Based Key Derivation Function 2) 2ª Função de Derivação de Chave Baseada em Senha, para armazenar senhas de forma segura com Salt aleatório (gerado para cada usuário), tamanho do hash configurável geralmente é de 256 bits ou 512 bits.

b) Proteção contra ataques de força bruta, limitando tentativas de login.

Utilizando o Identity framework implementamos uma medida de proteção contra ataques de força bruta, bloqueando o utilizador após 6 tentativas erradas da palavra-passe por 5 minutos.

c) Expiração automática da sessão do usuário após um período de inatividade.

Como mostrado no código a seguir foi configurado um construtor do Identity framework para se estabelecer o limite de tempo de inatividade de um utilizador, resultando no logout automático.

Exemplo:

```
builder.Services.ConfigureApplicationCookie(options =>
{
    options.ExpireTimeSpan = TimeSpan.FromMinutes(10); // Tempo total do cookie
    options.SlidingExpiration = false;
    options.Cookie.HttpOnly = true;
    options.LoginPath = "/Account/Login";
    options.AccessDeniedPath = "/Account/AccessDenied";

});
```

d) Bloqueio de múltiplas sessões, permitindo apenas um login por usuário por vez.

Foi implementado um bloqueio de múltiplas sessões utilizando o metodo `PasswordSignInAsync` e o método logout da página logout.razor.

e) Proteção contra XSS, SQL injection e DoS com Rate limiting

SQL injection

Entity Framework Core (EF Core): O EF Core usa parametrização nas consultas SQL, o que significa que ele não permite a injeção direta de SQL malicioso. Ao usar o método DbContext para acessar o banco de dados, ele gera as consultas SQL de maneira segura, evitando que dados maliciosos possam ser executados como comandos SQL.

XSS

Já tem proteção contra ataques XSS com binding Razor (@bind, @page, @model...) existentes da aplicação Blazor.

Exemplo:
```
<InputText @bind-Value="Input.Email" class="form-control" autocomplete="username" aria-required="true" placeholder="name@example.com" />
<label for="email" class="form-label">Email</label>
<ValidationMessage For="() => Input.Email" class="text-danger" />
```

DoS

Foi aplicado o framework RateLimiting do ASP.NET Core que lida com as requisições e bloqueia o IP do utilizador caso se exceda as 100 requisições em 1 minuto.

Exemplo:
```
builder.Services.Configure<IpRateLimitOptions>(builder.Configuration.GetSection("IpRateLimiting"));
builder.Services.AddSingleton<IRateLimitConfiguration, RateLimitConfiguration>();
```

2- Autenticação Multifator MFA

a) Implementação de MFA via código enviado por e-mail ou SMS.

Fez-se uma simulação de confirmação do e-mail após o registro do utilizador existe uma área para a colocação de uma conta 2FA com o Microsoft authenticator.

b) O código deve ser válido por um curto período.

Implementamos a duração de 1 minuto para os códigos de autenticação multifator com ligação do Microsoft authenticator.

3- Proteção contra acessos indevidos

a) Implementação de controle de acesso baseado em sessões, impedindo que usuários não autenticados acessem páginas restritas.

O Identity framework fornece uma medida de autenticação e autorização para utilizadores, utilizando o atributo `[Authorize]` e a tag `<AuthorizeView></AuthorizeView>`.

b) Proteção contra alteração direta da URL, garantindo que apenas usuários autorizados acessem determinados conteúdos.

O Atributo `[Authorize]` do Identity Framework faz o tratamento de redirecionar os utilizadores para as páginas permitidas ao seu acesso sem alteração da URL e acesso aos dados guardados.

4- Segurança de sessão e cookies

a) Uso de cookies seguros (Secure e HttpOnly) para evitar roubo de sessão.

Quando um utilizador faz login com o Identity, o servidor emite um cookie de autenticação (.AspNetCore.Identity.Application), esse cookie é armazenado no navegador do utilizador. Em cada requisição ao servidor, o cookie é enviado, permitindo que o servidor identifique o utilizador, isso funciona como uma sessão autenticada. Implementado no arquivo program.cs utilizando a estrutura “builder.Services.AddIdentityCore<ApplicationUser>(options =>{}”.

b) Regeneração do ID da sessão periodicamente para evitar sequestro de sessão.

No ficheiro PersistingRevalidatingAuthenticationStateProvider.cs encontra-se a regeneração periódica aplicada as sessões ativas.

5- Segurança de upload de arquivos

a) Permitir apenas upload de tipos de arquivos seguros.

A medida de segurança implementada foi não permitir a submissão de qualquer tipo de ficheiro.

b) Limitar o tamanho máximo do arquivo.

6- Restrição de Acesso por País(Opcional)

a) Bloquear ou permitir acesso à aplicação apenas para determinados países, com base no IP do usuário.

Não foi implementada na aplicação.

Conclusão

A autenticação segura é essencial para proteger o acesso ao sistema. Deve-se utilizar senhas fortes, armazenadas com algoritmos de hash seguros, evitar práticas vulneráveis como o armazenamento de senhas em texto puro. Além disso, o uso de HTTPS é obrigatório para evitar o vazamento de credenciais em trânsito.

A implementação de uma autenticação segura é a base da segurança de qualquer sistema, sendo necessária a combinação de criptografia robusta, a proteção contra acessos indevidos, aumenta significativamente a resiliência do sistema contra ataques automatizados e mal-intencionados, exigindo uma abordagem proativa com monitoramento contínuo.

Concluímos que todas as medidas de segurança, algoritmo, protocolo implementados em alguma aplicação é necessária para a confidencialidade e confiabilidade de todo sistema.