eHealth Application

Inês Santos
Mauro Filho
Patricia Cardoso

Trabalho realizado no âmbito da disciplina de
Segurança Informática e nas Organizações

DETI
Universidade de Aveiro
16 de novembro de 2022

Conteúdo

1 Introdução

Este projeto tem como objetivo desenvolver uma aplicação web para uma clínica de saúde suscetível a vulnerabilidades, e posteriormente uma versão segura da mesma.

2 A aplicação

A aplicação é constituída pela página principal, Home, que contém os serviços que a clínica disponibiliza, as especialidades a que se dedica, e os seus médicos. Para além disto, existe uma secção que permite ao utilizador entrar em contacto com a clínica. Esta secção denominada "Contact us"possui como campos, o nome do utilizador, o seu email, o assunto e a mensagem que pretende enviar. Existe ainda, uma página que permite marcar consultas, escolhendo-se qual o serviço pretendido, o médico e a data. No entanto, um utilizador só pode marcar consultas se já tiver efetuado o login. Para isso, existe a página "Log in to Patient Account". Uma vez feito o login, o utilizador é redirecionado para a página do seu perfil. Esta página contém o seu nome, o seu email e uma lista das suas consultas já agendadas. Para além disso, existem dois botões que permitem, fazer o download do resultado de um teste que o utilizador realizou (disponibilizado no servidor por um médico a partir da sua própria aplicação não desenvolvida neste projeto) e o outro botão para fazer log out.

3 Tecnologias

Toda a parte de servidor, e gestão de base de dados, da aplicação foi desenvolvida na linguagem Python. Para criar o servidor, utilizamos a biblioteca "CherryPy", e, para a gestão da base de dados, utilizamos a biblioteca "SQLite". Ambas nos permitem demonstrar tudo aquilo que é proposto no âmbito deste projeto, e por isso escolhemos trabalhar com as mesmas para o desenvolvimento do back-end aplicação.

Para o desenvolvimento do front-end, utilizámos as ferramentas base de desenvolvimento web: HTML, CSS e Javascript.

4 Vulnerabilidades

4.1 CWE-89: SQL Injection

Neste caso, o atacante injeta uma declaração SQL especial num campo de texto. Ao colocar sintaxe SQL em entradas controladas pelo utilizador, a query SQL gerada pode fazer com que o conteúdo dessas entradas seja interpretado como SQL em vez de dados comuns. Alterando a lógica da query pode ser possível ignorar verificações de segurança, mostrar informação confidencial e remover conteúdos da base de dados.

4.1.1 Aplicação não-segura:

Na aplicação não-segura desenvolvida para este projeto, utilizamos inputs dos utilizadores diretamente na escrita do comando SQL, concatenando este input ao comando pré escrito, permitindo assim que um atacante realize ações na nossa base de dados diferentes daquelas inicialmente propostas, ao manipular o comando. Um exemplo disso pode ser visto no login, visto que é possível alterar a query de tal forma a conseguir aceder ao profile sem providenciar uma palavra passe correta.

4.1.2 Avaliação:

SQL injection	Score: 10.0
Attack Vector:	Network
Attack Complexity:	Low
Privileges required:	None
User Interaction:	None
Scope:	Changed
Confidentiality:	High - É possível aceder a qualquer conta com apenas o seu email, pondo em risco todos os dados do paciente.
Integrity:	High - Acesso completo à conta permite que o atacante consulte e altere dados do paciente.
Availability:	None

4.1.3 Aplicação segura:

Para prevenir que isso aconteça, implementamos "prepared statements", para realizar queries parametrizadas na versão segura da aplicação. Dessa forma, o atacante perde o poder de modificar a query, e conseguimos garantir a integridade das mesmas

4.2 CWE-79: Cross-site Scripting

Estes ataques acontecem quando são injetados scripts maliciosos em sites confiáveis. As falhas que permitem este tipo de ataque apenas implicam que exista um web request. Após o conteúdo entrar na aplicação,é gerada uma página que contém a informação não confiável. Durante a geração da página,a aplicação não consegue prevenir a informação que é executada pelo navegador como por exemplo HTML e JavaScript. Neste momento,a vítima visita a página não confiável e dados como cookies e outras informações confidenciais retidas pelo navegador podem ser acedidos.

4.2.1 Aplicação não segura:

Na aplicação não-segura desenvolvida para este projeto, é possível que um atacante injete um script no campo "date"da tabela "Bookings"da base de dados ao realizar uma marcação diretamente através do url (sem passar pelo formulário presente na página web). Ao inserir o ID de um utilizador no url, é possível que o atacante injete este script malicioso no perfil de outra pessoa, que será executado quando esta fizer log-in. Desta forma, qualquer utilizador por ser alvo direto de um ataque.

4.2.2 Avaliação:

XSS	Score: 8.3
Attack Vector:	Network
Attack Complexity:	Low
Privileges required:	None
User Interaction:	None
Scope:	Changed
Confidentiality:	Low - O script será executado pelo browser de outro utilizador, sendo possível que aceda a dados confidenciais.
Integrity:	High - É possível realizar ações nas conta de outros utilizadores sem ter acesso a esta, injetando código que é executado quando a vítima visita a página de perfil.
Availability:	Low - O código executado pode afetar a disponibilidade de certos serviços, como a marcação de consultas

4.2.3 Aplicação segura:

Na aplicação segura, adicionamos uma checagem de input no campo "date"da tabela "bookings", que rejeita qualquer marcação com uma data em um formato inválido. Tornando assim, impossível que um atacante injete um script neste campo ao invés de uma data.

4.3 CWE-352: Cross-Site Request Forgery (CSRF)

A aplicação não verifica corretamente se um pedido foi enviado pelo utilizador. Assim, quando um servidor recebe um pedido sem possuir nenhum mecanismo para verificar a intencionalidade desse pedido, o atacante pode fazer com que o cliente envie um pedido não intencional. Isto pode ser feito através do url ou do carregamento de imagem. Pode resultar em exposição de dados.

4.3.1 Aplicação não segura:

Na aplicação não-segura, é possível que um atacante faça uma marcação diretamente por um URL, como já foi descrito anteriormente. Ao inserir o ID de um utilizador no URL, é possível que o atacante faça esta marcação em nome de outro utilizador qualquer, dado que o ID é o único método de relacionar uma marcação a um utilizador, e este é enviado como texto para a aplicação cliente.

4.3.2 Avaliação:

CSRF	Score: 7.2
Attack Vector:	Network
Attack Complexity:	Low
Privileges required:	None
User Interaction:	None
Scope:	Changed
Confidentiality:	None
Integrity:	Low - É possível realizar ações na conta de outros utilizadores sem ter acesso a esta
Availability:	Low - Pode-se utilizar as contas de outros utilizadores para ocupar vagas para consulta, impedindo outros utilizadores de marcar consultas na mesma data.

4.3.3 Na aplicação segura:

Na aplicação segura, adicionamos um fator extra de identificação na submissão de marcações, o que torna obrigatória a submissão através do formulário web, sem ser possível mais realizar marcações diretamente por urls.

Ao fazer signup, é criado no servidor um par de chaves de cifra (pública e privada) pessoal para cada utilizador. Esta chave é utilizada para cifrar o email do utilizador, que depois é enviado para a aplicação cliente. Ao realizar uma marcação, este email cifrado é enviado de volta para o servidor, que irá decifrá-lo e confirmar a identidade do utilizador a realizar a marcação. Neste caso, se torna impossível para o atacante realizar uma marcação em nome de outro utilizador, pois este nunca terá acesso ao email cifrado que é enviado no login de cada utilizador.

4.4 CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

A aplicação utiliza input do utilizador para gerar um path para guardar um ficheiro. Esse input não é devidamente verificado, permitindo a que se guarde o ficheiro fora do diretório onde é suposto, o que pode levar à perda de dados e ficheiros importantes.

4.4.1 Aplicação não-segura:

Quando uma mensagem é submetida pelo formulário da página inicial, esta é salva no servidor como um ficheiro de texto dentro do diretório "Contacts". O nome deste ficheiro é o título da mensagem que foi inserido pelo utilizador, utilizado diretamente, sem nenhum tipo de tratamento. Desta forma, é possível que um atacante faça a submissão de um ficheiro para qualquer diretório do servidor. Por exemplo, é possível excluir totalmente a nossa base de dados, se o atacante enviar uma mensagem com o título "../Database.db".

4.4.2 Avaliação:

Path Traversal	Score: 10.0
Attack Vector:	Network
Attack Complexity:	Low
Privileges required:	None
User Interaction:	None
Scope:	Changed
Confidentiality:	High - Se o novo ficheiro substituir um dos ficheiros de código já existentes, será possível executar código no servidor, possivelmente acedendo a dados confidenciais
Integrity:	High - Qualquer ficheiro do servidor pode ser alterado
Availability:	High - É possível escrever por cima da base de dados ou qualquer outro ficheiro necessário ao funcionamento da aplicação

4.4.3 Aplicação segura:

Na aplicação segura, os dados inseridos pelo utilizador já não interferem mais com o path onde o ficheiro da sua mensagem será guardada. Ao invés disso, o nome do ficheiro é gerado pelo servidor de maneira única, ao utilizar o exato momento em que a mensagem chega ao servidor como nome deste ficheiro. Como é impossível que duas mensagens sejam processadas exatamente a mesma hora no servidor, também evitamos que os ficheiros se sobreponham.

4.5 CWE-798: Use of Hard-coded Credentials

A aplicação protege os seus dados com recurso a cifras mas as suas chaves e/ou passwords encontram-se diretamente guardadas no código. Chaves estáticas levam a que mais dados sejam comprometidos caso um atacante as consiga obter.

4.5.1 Aplicação não-segura:

Na aplicação não-segura, ciframos o nome dos utilizadores registados na aplicação utilizando um único par de chaves, salvas em ficheiros .pem, protegidos por uma senha que está hard-coded no ficheiro "DatabaseFunctions.py". Uma clara ameaça à segurança da aplicação, já que o atacante poderá decifrar todos os dados que conseguir caso obtenha as chaves e a senha.

4.5.2 Avaliação:

Hard-Coded Credentials	Score: 5.8
Attack Vector:	Network
Attack Complexity:	Low
Privileges required:	None
User Interaction:	None
Scope:	Changed
Confidentiality:	Low - Caso o atacante obtenha as chaves poderá decifrar todos os dados que consiga obter
Integrity:	None
Availability:	None

4.5.3 Aplicação segura:

Na aplicação segura, cada utilizador possui um par de chaves pessoal, que está protegido pela sua própria palavra passe, que por sua vez está guardada na base de dados. Desta forma, conseguimos retirar a passe do código e ainda limitar o acesso de um atacante aos dados do servidor caso venha a ter acesso à alguma chave, já que cada utilizador tem a sua própria chave que apenas pode decifrar suas próprias informações.