Aviso legal

Serviços para Aplicações Web e Móveis

Segurança

Fábio Marques (fabio@ua.pt)
Mestrado em Informática Aplicada

Escola Superior de Tecnologia e Gestão de Águeda
Universidade de Aveiro

Ao construir uma aplicação Web devemos assumir que:

- Vai ser atacada
- Os dados vão ser roubados
- Vai existir fuga de informação
- Existe um perigo eminente todos dias

- Injection
- Quebras de autenticação e gestão de sessões
- Utilização de Scripts cruzados (Cross-Site Scripting - XSS)
- Pedidos Externos (Cross-Site Request Forgery - CSRF)

- Ocorre quando um utilizador envia ataques baseados em texto, para explorar a sintaxe do interpretador de destino.
- Tem como principal consequências: perda ou corrupção de dados, bloqueio de acesso ao proprietário dos dados, etc
- Um invasor poderá ser qualquer pessoa que tenha permissões de envio de dados não confiáveis ao sistema.
- Embora não seja caso único, o exemplo mais conhecido deste tipo de ataque é o SQL Injection

Ocorre quando um invasor utiliza fugas ou falhas nas funções de autenticação ou de gestão de sessões para se fazer passar por outros.

Três cenários e respetivas soluções:

1. Password guardada em texto simples
   Criar um hash da password, o que pode ser feito com um módulo como o bcrypt
2. ID de sessão no URL
   Utilizar cookies para guardar a sessão no cliente
3. Acesso de scripts às cookies no cliente
   Indicar ao cliente que a cookie apenas poderá ser acedida em determinadas condições

Um ataque deste tipo acontece quando o invasor envia scripts baseados em texto que exploram o interpretador do browser.

1. Ataques não persistentes
   Este tipo de ataque consiste em injetar scripts enviados para o browser de um utilizador por solicitação do próprio
   A primeira solução é validar, proteger e higienizar
2. Ataques persistentes
   Neste tipo de ataque o script é permanentemente armazenado na base de dados da aplicação.
   Informar o browser de que apenas deve permitir a manipulação de conteúdo proveniente de fontes confiáveis

- Os ataques do tipo CSRF (Cross-Site Request Forgery) forçam o utilizador final a executar ações indesejadas numa aplicação onde se encontram autenticados
- Os ataques CSRF visam, especificamente, alterações de estado e não o roubo de dados
- Este tipo de ataque ocorre quando um site permite que as alterações sejam feitas através de comandos GET do protocolo HTTP

A solução é não permitir que as alterações ocorram em pedidos GET.

- é o processo de verificação da identidade de um utilizador do sistema
- garante que o utilizador que está a aceder ao sistema ou aplicação é quem diz ser (ou pelo menos está na posse da informação de autenticação correta)
- existem vários métodos de Autenticação
  - baseados em conhecimento, como senhas ou PINs
  - multi fator (por exemplo, senha e autenticação biométrica)
  - baseados em certificado, como certificados digitais
  - biométrica, como impressões digitais ou reconhecimento facial
  - baseados em token, como cartões inteligentes ou tokens de segurança
  - …

- Middleware para o Node
- Facilita a implementação da autenticação e da autorização
- Tem um conjunto de características como o Single sign-on com OpenID[1] e OAuth, login/password[2], suporte de sessões persistentes, etc
- Tem mais de 500 estratégias de autenticação que podem ser suportadas pela aplicação

```
npm install passport /* instalacao do middleware */
npm install passport-local /* estrategia login/password */
npm install passport-google-oidc /* estrategia OpenID Connect  */
```

[1] https://www.passportjs.org/packages/passport-google-oidc/
[2] https://www.passportjs.org/tutorials/password/
[3] https://www.passportjs.org

- o processo para determinar quais as ações ou recursos que um utilizador tem permissão para aceder ou executar após a autenticação
- define e impõe políticas de controlo de acesso com base na identidade, funções ou outros atributos do utilizador
- garante que os utilizadores apenas podem aceder a recursos e executar ações que lhes são explicitamente permitidas
- ajuda a aplicar políticas de segurança, proteger dados confidenciais e evitar atividades não autorizadas no sistema
- é um componente crítico da segurança de um sistema e deve ser implementado de forma robusta e eficaz

Duas das abordagens mais comuns para a autorização são:

- Baseada em funções (role): os utilizadores são atribuídos a um ou mais grupos de acordo com as suas funções
- Baseada em regras (rule): os utilizadores são atribuídos a um ou mais grupos de acordo regras específicas

```
// Authorization middleware
function authorize(req, res, next) {
  if (!req.user) {
    return res.status(401).json({ error: "Unauthorized" });
  }
  // Check if user has the required role
  if (!req.user.roles.includes("admin")) {
    return res.status(403).json({ error: "Forbidden" });
  }
  // User is authorized, proceed to the next middleware or route handler
  next();
}

// Example route protected by authorization middleware
app.get("/admin/dashboard", authorize, (req, res) => {
  res.send("Welcome to the admin dashboard!");
});
```