Instituto Politécnico de Setúbal

Escola Superior de Tecnologia de Setúbal

A01:2021 – Broken Access Control

Segurança de Informação e de Software - MES

João Silva Gomes 201501755

OWASP

O “Open Web Application Security Project”, ou OWASP, é uma organização internacional sem fins lucrativos e Open Source dedicada à segurança de aplicações web.

Como Open Source os seus princípios fundamentais são é que todos os seus materiais estejam disponíveis a todos e que sejam e sejam facilmente acessíveis.

O seu projeto mais conhecido é o OWASP Top 10.

OWASP Top 10

O OWASP Top 10 é um relatório atualizado regularmente que descreve questões de segurança para aplicações web, com foco nos 10 riscos mais críticos.

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

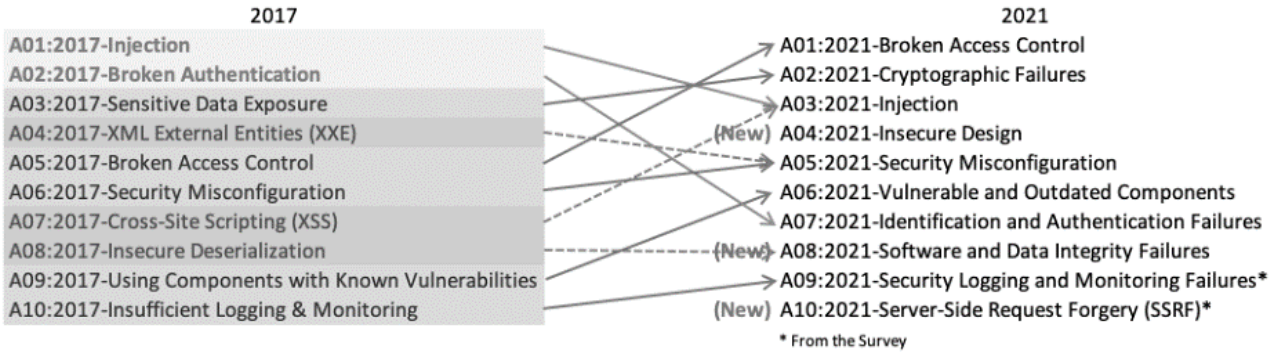
https://github.com/OWASP/Top10

A01:2021 – Broken Access Control

CWEs Mapped	Max Incidence Rate	Avg Incidence Rate	Avg Weighted Exploit	Avg Weighted Impact	Max Coverage	Avg Coverage	Total Occurrences	Total CVEs
34	55.97%	3.81%	6.92	5.93	94.55%	47.72%	318,487	19,013

O “Access control”impõe políticas de forma que os utilizadores não possam agir fora das permissões pretendidas.

As falhas normalmente levam à divulgação não autorizada de informações, modificação ou destruição dos dados ou à execução de uma funcionalidade fora dos limites do utilizador.



OWASP Top 10 - 2013		OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

Termos

Autenticação e Autorização (Authentication Authorization)

A "Authentication" é a prática de segurança de confirmar que alguém é quem afirma ser, enquanto a "Authorization"é o processo de determinar qual nível de acesso é concedido a cada usuário.

Quando um Utilizador acede ao seu e-mail ou portal bancário on-line, este uma combinação de login e senha que somente ele deve saber. O software usa essas informações para autenticar o Utilizador.
Algumas aplicações têm requisitos de autorização muito mais rígidos do que outros; embora uma senha seja suficiente para alguns, outros podem exigir autenticação de dois fatores ou confirmação biométrica, como impressão digital ou digitalização de identificação facial.

Uma vez autenticado, um Utilizador só poderá ver as informações que está autorizado a acessar. No caso de uma conta bancária online, o utilizador só pode ver informações relacionadas com a sua conta bancária pessoal.
Entretanto, um gestor de fundos do banco pode iniciar sessão na mesma aplicação e ver dados sobre as participações financeiras globais do banco.

Controle de Acesso (Access Control)

O Controle de Acesso refere-se a um sistema que controla o acesso a informações ou funcionalidades.

Controle de acesso é um termo de segurança usado para se referir a um conjunto de políticas para restringir o acesso a informações ou funcionalidades.

O Controle de Acesso determina se o Utilizador tem permissão para executar a ação.

Authorization vs. Access Control

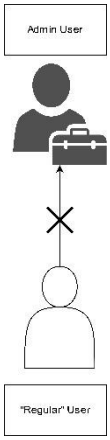
Se a Autorização envolve a definição de uma política, o Controle de Acesso coloca as políticas em funcionamento.

Esses dois termos não são sinonimos mas complementação.

Tipos de Controlo de Acesso

1. Vertical Access Control

Os Controlos de Acesso Vertical são mecanismos que restringem o acesso a funcionalidades confidenciais a tipos específicos (Roles) de Utilizadores.



Com Controlos de Acesso Vertical, diferentes tipos de Utilizadores têm acesso a diferentes funções do aplicativo.

Exemplo:

Um administrador pode modificar ou eliminar a conta de qualquer Utilizadore, enquanto um Utilizador comum não tem acesso a essas ações.

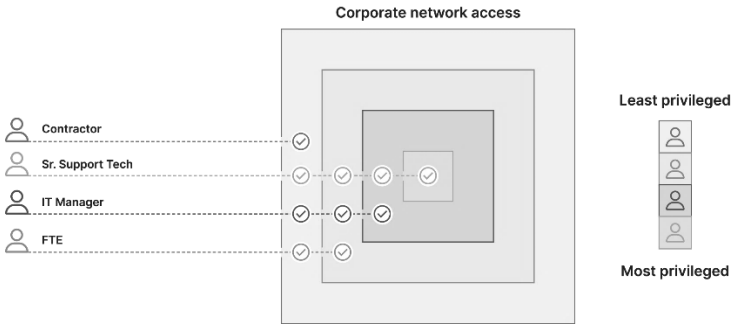
Os Controlos de Acesso Vertical podem ser implementações mais refinadas de modelos de segurança projetados para impor políticas de negócios, como separação de funções e privilégios mínimos.(Separation of Duties and Principle of Least Privilege)

1.1 Separation of Duties and Principle of Least Privilege

1.1.1 **Principle of Least Privilege**: Os Utilizadores devem ter apenas o mínimo de privilégios necessários para realizar as suas tarefas e nada mais. Isso reduz a exploração de autorização limitando o acesso aos diversos recursos da aplicação

Exemplo:

Um profissional de marketing precisa de acesso ao CMS do Website da sua organização para adicionar e atualizar o conteúdo.Mas se também lhe for dado acesso ao Source Code - que não é necessária para atualizar o conteúdo - o impacto negativo da sua conta for comprometida pode ser muito maior.



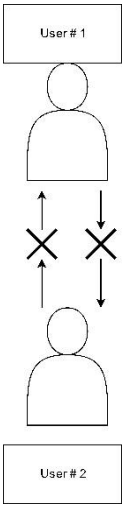
1.1.2 **Separation of Duties**: Além de limitar o nível de privilégio dos Utilizadores, também deve-se limitar as tarefas do Utilizador ou os trabalhos específicos que estes podem executa.

Nenhum Utilizador deve ser responsável por mais de uma função relacionada.

Exemplo:

A mesma pessoa não deve ser responsável pelo desenvolvimento e teste de um sistema de segurança.

2. Horizontal Access Controls



Os Controlos de Acesso Horizontal são mecanismos que restringem o acesso a recursos a usuários específicos.

Com controles de acesso horizontais, diferentes usuários têm acesso a um subconjunto de recursos do mesmo tipo.

Exemplo:

Uma plataforma bancária permitirá que um Utilizador visualize transações e faça pagamentos das suas próprias contas, mas não das contas de qualquer outro Utilizador.

3. Context-dependent access controls

Os Context-dependent access controls restringem o acesso à funcionalidade e aos recursos com base no estado da aplicação ou na interação do utilizador com a mesma, impedindo que um utilizador execute acções na ordem errada.

Broken Access Control

Broken access controls permitem que os ataques contornem a autorização e executem tarefas como se fossem utilizadores privilegiados, como por exemlpo administradores. Por exemplo, uma aplicação Web pode permitir que um utilizador altere a conta com a qual tem sessão iniciada simplesmente alterando parte de um URL, sem qualquer outra verificação

Exemplos:

1. **Vertical Privilege Escalation**

Ocorre quando um atacante obtém acesso a funcionalidades privilegiadas às quais não está autorizado a aceder.

Na sua forma mais básica, Vertical Privilege Escalation surge quando uma aplicação não impõe qualquer proteção para a funcionalidade sensível (funcionalidade desprotegida).

Por exemplo, as funções administrativas podem estar ligadas a partir da página login de um administrador, mas não a partir da página login de um utilizador.

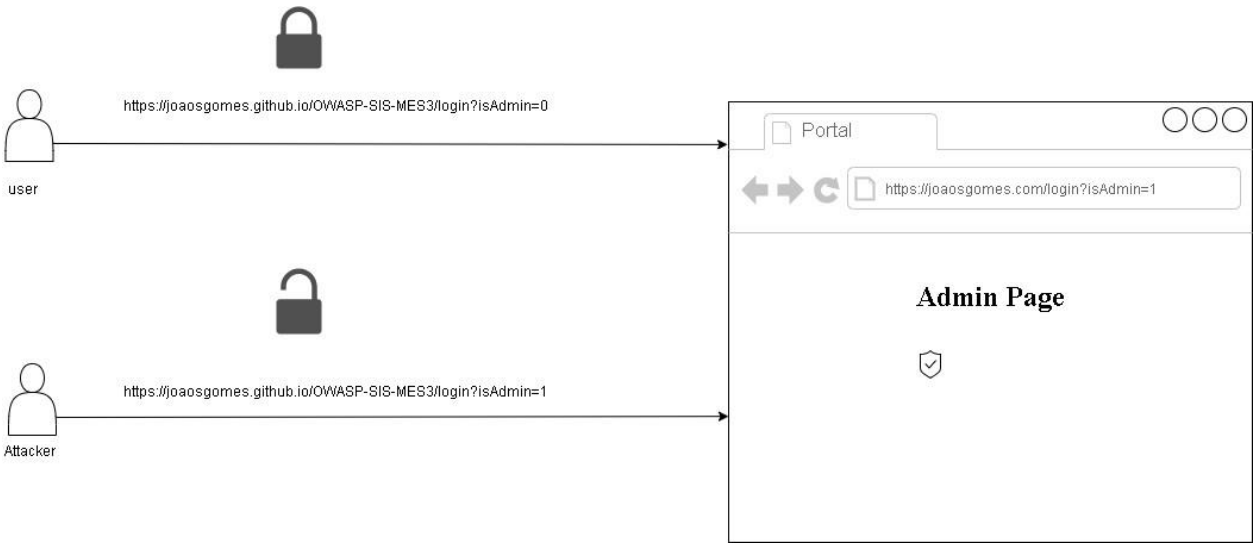
No entanto, um utilizador poderá ter acesso às funções administrativas navegando para o URL de administração relevante.

Exemplo:

Um WebSite pode alojar funcionalidades sensíveis no seguinte URL (A aplicação toma decisões de controlo de acesso com base no valor passado por query):

https://joaosgomes.github.io/OWASP-SIS-MES3/login?isAdmin=0

Esta abordagem é insegura porque um utilizador pode modificar o valor e aceder a funcionalidades para as quais não está autorizado, tais como funções administrativas.



Como Detectar:

Isto pode ser acessível a qualquer utilizador, não apenas aos utilizadores administrativos que têm uma ligação à funcionalidade na sua interface de utilizador. Em alguns casos, o URL administrativo pode ser divulgado noutros locais, como o ficheiro robots.txt:

https://joaosgomes.github.io/OWASP-SIS-MES3/robot.txt

Mesmo que o URL não seja divulgado em lado nenhum, um atacante pode ser capaz de utilizar uma lista de palavras para forçar a localização da funcionalidade sensível.

Em alguns casos, a funcionalidade sensível é ocultada dando-lhe um URL menos previsível.

Este é um exemplo da chamada "segurança por obscuridade". No entanto, a ocultação da funcionalidade sensível não proporciona um controlo de acesso eficaz, porque os utilizadores podem descobrir o URL ofuscado de várias formas.

Aplicação pode ainda divulgar o URL aos utilizadores. [Inspect Sorce Code of Page]

2. Horizontal Privilege Escalation

Ocorre quando um atacante obtém acesso a recursos pertencentes a outro utilizador com o mesmo nível de privilégios.

Horizontal Privilege Escalation ocorre quando um utilizador consegue aceder a recursos pertencentes a outro utilizador, em vez de aceder aos seus próprios recursos desse tipo.

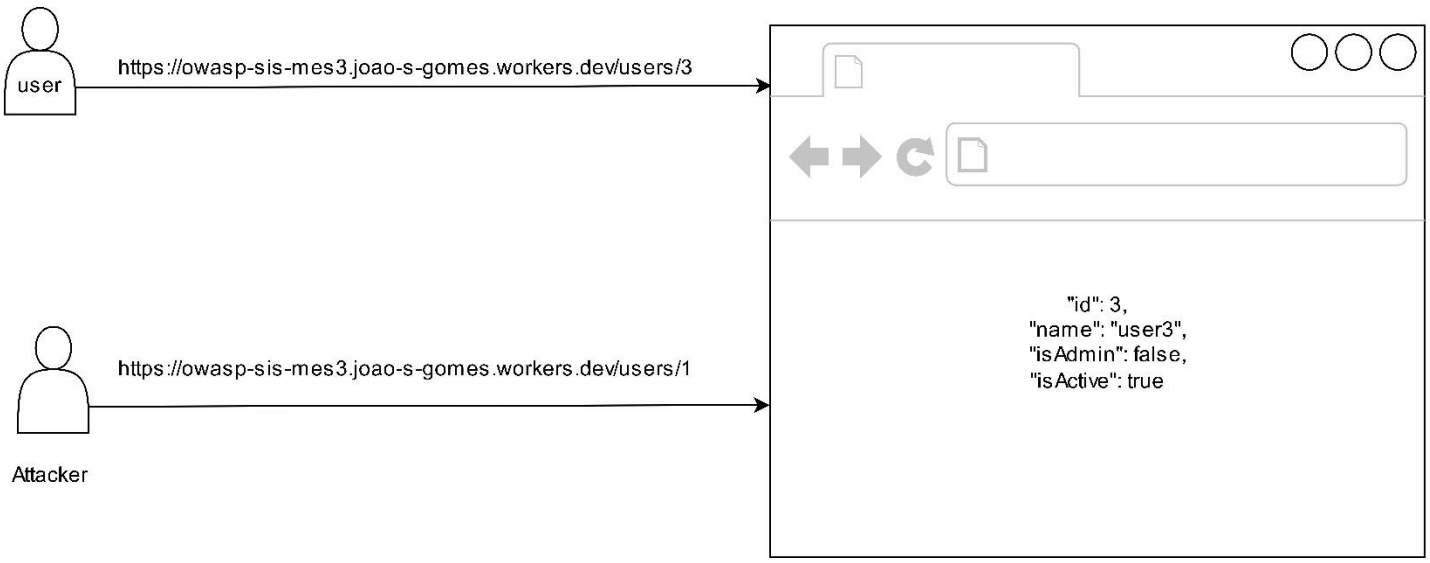
Por exemplo, se um Utilizador puder aceder aos registos de outros Utilizadores, bem como aos seus próprios, trata-se de Horizontal Privilege Escalation

Os ataques de Horizontal Privilege Escalation podem utilizar tipos de métodos de exploração semelhantes aos de Vertical Privilege Escalation.

Exemplo:

https://owasp-sis-mes3.joao-s-gomes.workers.dev/users/3

Se um atacante modificar o valor do parâmetro id para o de outro utilizador, poderá obter acesso à página da conta de outro utilizador e aos dados e funções associados.



Como Prevenir:

Em algumas aplicações, o parâmetro explorável não tem um valor previsível.

Por exemplo, em vez de um número identificador crescente, uma aplicação pode utilizar identificadores globalmente únicos (GUIDs) para identificar utilizadores.

Isto pode impedir um atacante de adivinhar ou prever o identificador de outro utilizador.

Exemplo:

https://owasp-sis-mes3.joao-s-gomes.workers.dev/usersUUID/f31dcf8d-1ff6-4033-a5bf-3835511c6341

3. **Horizontal to vertical privilege escalation**

Muitas vezes, um ataque de escalada de privilégios horizontal pode ser transformado numa escalada de privilégios vertical, comprometendo um utilizador mais privilegiado.

(Admin)

https://owasp-sis-mes3.joao-s-gomes.workers.dev/users/1

Se o utilizador alvo for um administrador, o atacante terá acesso a uma página de conta administrativa. Esta página pode revelar a palavra-passe do administrador ou fornecer um meio de a alterar, ou pode fornecer acesso direto a funcionalidades privilegiadas.

Outros

Accessing API with missing access controls for POST, PUT and DELETE.

https://github.com/joaosgomes/OWASP-SIS-MES3/blob/main/index.js#L303 (Enpoint deleteUser)

https://github.com/joaosgomes/OWASP-SIS-MES3/blob/main/index.js#L328 (Enpoint deleteUserSecure)

CORS misconfiguration allows API access from unauthorized/untrusted origins.

CORS - Cross-Origin Resource Sharing (Compartilhamento de recursos com origens diferentes) é um mecanismo que usa cabeçalhos adicionais HTTP para informar a um navegador que permita que um aplicativo Web seja executado em uma origem (domínio) com permissão para acessar recursos selecionados de um servidor em uma origem distinta. Um aplicativo Web executa uma requisição cross-origin *HTTP ao solicitar um recurso que tenha uma origem diferente (domínio, protocolo e porta) da sua própria origem.*

Browser Error:

Access to fetch at 'https://owasp-sis-mes3.joao-s-gomes.workers.dev/statusnocors' from origin 'https://joaosgomes.github.io' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource. If an opaque response serves your needs, set the request's mode to 'no-cors' to fetch the resource with CORS disabled.

Ver:

https://joaosgomes.github.io/OWASP-SIS-MES3/

https://owasp-sis-mes3.joao-s-gomes.workers.dev/statusnocors

https://owasp-sis-mes3.joao-s-gomes.workers.dev/status

Impacto das Vulnerabilidades de Controlo de Acesso

1. Acesso não autorizado a Aplicações:
   1.1. Confidentiality – Acesso aos dados/informação de outros Utilizadores.
   1.2. Integrity – Acesso para atualizar dados/informação.
   1.3. Availability – Acesso para eliminar dados/informação.
2. Às vezes pode ser encadeado com outras vulnerabilidades

Enforce Access Controls

https://owasp.org/www-project-proactive-controls/v3/en/c7-enforce-access-controls

1. Design Access Control Thoroughly Up Front
2. Force All Requests to Go Through Access Control Checks
3. Deny by Default
4. Principle of Least Privilege
5. Don’t Hardcode Roles
6. Log All Access Control Events

Referencias:

https://owasp.org/Top10/A01_2021-Broken_Access_Control/
https://www.cloudflare.com/learning/security/threats/owasp-top-10/
https://www.cloudflare.com/learning/access-management/what-is-access-control/
https://www.cloudflare.com/learning/access-management/authn-vs-authz/
https://www.cloudflare.com/learning/access-management/principle-of-least-privilege/
https://owasp.org/www-project-proactive-controls/v3/en/c7-enforce-access-controls
https://portswigger.net/web-security/access-control
https://github.com/joaosgomes/OWASP-SIS-MES3 (Estudo Prático)