

Code:

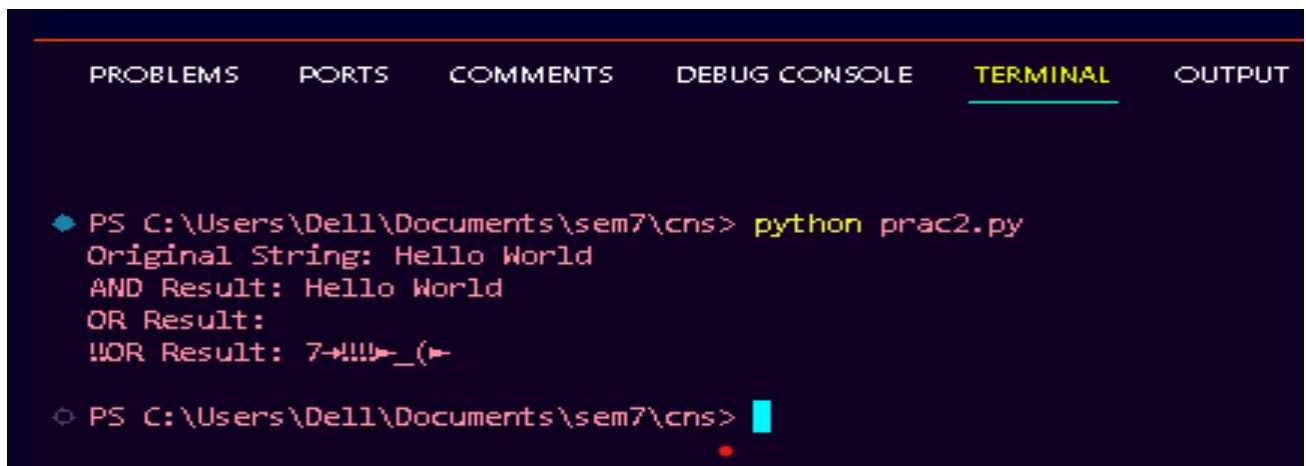
Write a program that combines a string with a values "Hello World " The Program should and and Xor Or Each character in this string with 127 end display result

```
input_string = "Hello World"
and_result = ""
or_result = ""
xor_result = ""

for char in input_string:
    # Perform AND operation with 127 on each character
    and_result += chr(ord(char) & 127)
    # Perform OR operation with 127 on each character
    or_result += chr(ord(char) | 127)
    # Perform XOR operation with 127 on each character
    xor_result += chr(ord(char) ^ 127)

print("Original String:", input_string)
print("AND Result:", and_result)
print("OR Result:", or_result)
print("XOR Result:", xor_result)
```

Output:



```
PROBLEMS  PORTS  COMMENTS  DEBUG CONSOLE  TERMINAL  OUTPUT

◆ PS C:\Users\Dell\Documents\sem7\cns> python prac2.py
Original String: Hello World
AND Result: Hello World
OR Result:
XOR Result: 7-4!!!>_(<

○ PS C:\Users\Dell\Documents\sem7\cns>
```


Code :

symmertric key encryption alforithm using feistal block cipher structure

Python program to demonstrate

Feistel Cipher Algorithm

```
import binascii
```

Random bits key generation

```
def rand_key(p):
```

```
    import random
```

```
    key1 = ""
```

```
    p = int(p)
```

```
    for i in range(p):
```

```
        temp = random.randint(0,1)
```

```
        temp = str(temp)
```

```
        key1 = key1 + temp
```

```
    return(key1)
```

Function to implement bit exor

```
def exor(a,b):
```

```
    temp = ""
```

```
    for i in range(n):
```

```
        if (a[i] == b[i]):
```

```
            temp += "0"
```

else:

temp += "1"

return temp

Defining BinarytoDecimal() function

def BinaryToDecimal(binary):

Using int function to convert to

string

string = int(binary, 2)

return string

Feistel Cipher

PT = "Hello"

print("Plain Text is:", PT)

Converting the plain text to

ASCII

PT_Ascii = [ord(x) for x in PT]

Converting the ASCII to

8-bit binary format

PT_Bin = [format(y,'08b') for y in PT_Ascii]

PT_Bin = "".join(PT_Bin)

n = int(len(PT_Bin)//2)

L1 = PT_Bin[0:n]

R1 = PT_Bin[n::]

m = len(R1)

Generate Key K1 for the

first round

K1= rand_key(m)

Generate Key K2 for the

second round

K2= rand_key(m)

first round of Feistel

f1 = exor(R1,K1)

R2 = exor(f1,L1)

L2 = R1

Second round of Feistel

f2 = exor(R2,K2)

R3 = exor(f2,L2)

L3 = R2

Cipher text

bin_data = L3 + R3

str_data = ' '

for i in range(0, len(bin_data), 7):

slicing the bin_data from index range [0, 6]

and storing it in temp_data

temp_data = bin_data[i:i + 7]

passing temp_data in BinarytoDecimal() function

to get decimal value of corresponding temp_data

decimal_data = BinaryToDecimal(temp_data)

```

# Decoding the decimal value returned by
# BinarytoDecimal() function, using chr()
# function which return the string corresponding
# character for given ASCII value, and store it
# in str_data
str_data = str_data + chr(decimal_data)

print("Cipher Text:", str_data)

# Decryption
L4 = L3
R4 = R3

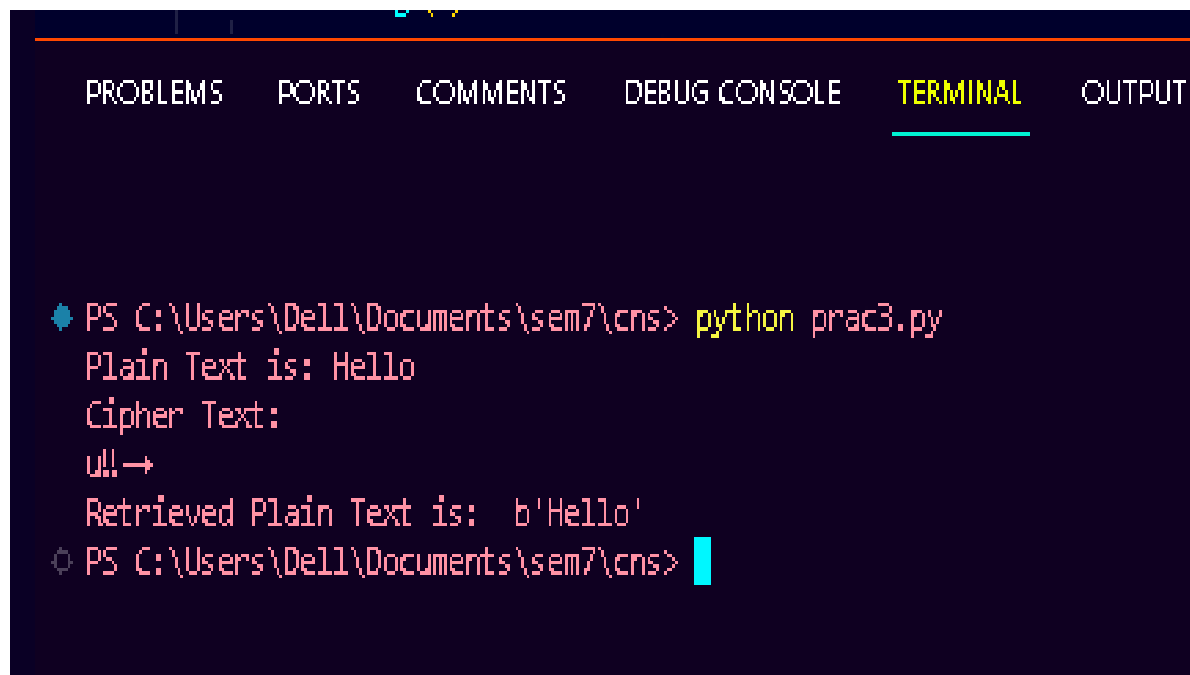
f3 = exor(L4,K2)
L5 = exor(R4,f3)
R5 = L4

f4 = exor(L5,K1)
L6 = exor(R5,f4)
R6 = L5
PT1 = L6+R6

PT1 = int(PT1, 2)
RPT = binascii.unhexlify( '%x'% PT1)
print("Retrieved Plain Text is: ", RPT)

```

Output:



The image shows a screenshot of a Visual Studio Code (VS Code) terminal window. The terminal has a dark background with a light blue border. At the top, there is a tab bar with several tabs: 'PROBLEMS', 'PORTS', 'COMMENTS', 'DEBUG CONSOLE', 'TERMINAL' (which is active and highlighted with a red underline), and 'OUTPUT'. The terminal content shows a PowerShell prompt 'PS C:\Users\Dell\Documents\sem7\cns>' followed by the command 'python prac3.py'. The output of the script is displayed in the following lines: 'Plain Text is: Hello', 'Cipher Text:', 'u!!→', and 'Retrieved Plain Text is: b'Hello''. The terminal ends with another PowerShell prompt 'PS C:\Users\Dell\Documents\sem7\cns>' and a red cursor.

```
PS C:\Users\Dell\Documents\sem7\cns> python prac3.py
Plain Text is: Hello
Cipher Text:
u!!→
Retrieved Plain Text is: b'Hello'
PS C:\Users\Dell\Documents\sem7\cns>
```


Code:

```
# RSA
```

```
import random
```

```
import math
```

```
#select 2 large prime numbers
```

```
def generate_p_and_q():
```

```
#Calculating 1 to 100 prime numbers
```

```
numbs = [i for i in range(2,101)]
```

```
for n in range(2,101):
```

```
    for i in range(2,math.ceil(n/2)+1):
```

```
        if n % i == 0:
```

```
            numbs.remove(n)
```

```
            break
```

```
        else:
```

```
            continue
```

```
#Selecting any 2 prime numbers randomly
```

```
p = random.choice(numbs)
```

```
numbs.remove(p)
```

```
q = random.choice(numbs)
```

```
return p, q
```

```
p,q = generate_p_and_q()
```

```
print(f'[+] p = {p} and q = {q}')
```

```
n = p * q
```

```
phi = (p - 1) * (q - 1)
```

```
print(f'[+] n = {n} and euler totient = {phi}')
```

```
#Calculating e -> gcd(e,phi) = 1 and 1 < e < phi.
```

```
def generate_e(phi):
```

```
    possible_e_values = []
```

```
    for i in range(2,phi):
```

```
        if math.gcd(i,phi) == 1:
```

```
            e=i
```

```
            possible_e_values.append(e)
```

```
    # print(possible_e_values)
```

```
    return random.choice(possible_e_values)
```

```
e = generate_e(phi)
```

```
print(f'[+] e = {e}')
```

```
def generate_d(e,phi):
```

```
    # d_list = []
```

```
    for i in range(2,phi):
```

```
        if (i*e) % phi == 1: # ed mod(phi) = 1
```

```

        d = i # As every unique public key have only one unique private key.
        # d_list.append(d)
        break

    # print(d_list)
    return d

d = generate_d(e,phi)

print(f'[+] d = {d}')
```

Message should be less than n (msg < n)

```

msg = random.randint(1,n)

print(f'[+] msg : {msg}')
```

```

def encrypt(msg,e,n):  #(msg^e) mod n
    c = pow(msg,e,n)
    return c

e_msg = encrypt(msg,e,n)

print(f'[+] Encrypted msg : {e_msg}')
```

```

def decrypt(msg,d,n):  #(msg^d) mod n
    p = pow(msg,d,n)
    return p

d_msg = decrypt(e_msg,d,n)

print(f'[+] Decrypted msg : {d_msg}')
```

Output:

```
PROBLEMS  PORTS  COMMENTS  DEBUG CONSOLE  TERMINAL  OUTPUT

◆ PS C:\Users\Dell\Documents\sem7\cns> python prac3.py
Plain Text is: Hello
Cipher Text:
u!!→
Retrieved Plain Text is: b'Hello'
◆ PS C:\Users\Dell\Documents\sem7\cns> python prac4.py
[+] p = 13 and q = 2
[+] n = 26 and euler totient = 12
[+] e = 5
[+] d = 5
[+] msg : 4
[+] Encrypted msg : 10
[+] Decrypted msg : 4
○ PS C:\Users\Dell\Documents\sem7\cns> 
```

Code:

```
from Crypto.Cipher import DES
from Crypto.Random import get_random_bytes

# Initialize the DES cipher with a random 8-byte key
key = get_random_bytes(8)
cipher = DES.new(key, DES.MODE_ECB)

# Your plaintext (must be a multiple of 8 bytes)
plaintext = b'Hello123'

print("Plaintext :", plaintext)

# Encryption
ciphertext = cipher.encrypt(plaintext)
print("Ciphertext:", ciphertext)

# Decryption
decipher = DES.new(key, DES.MODE_ECB)
decrypted_text = decipher.decrypt(ciphertext)
print("Decrypted Text:", decrypted_text.decode('utf-8'))
```

OUTPUT:

```
PROBLEMS 9 PORTS COMMENTS DEBUG CONSOLE TERMINAL OUTPUT

PS C:\Users\Dell\Documents\sem7\cns> python prac40.py
Plaintext : b'Hello123'
Ciphertext: b'\x0e\xfd\x0e\xcc\x91\n\x11\xda'
Decrypted Text: Hello123
PS C:\Users\Dell\Documents\sem7\cns>
```

Code

```
: import random
```

```
# public keys are taken
```

```
# p is a prime number
```

```
# g is a primitive root of p
```

```
p = int(input('Enter a prime number : '))
```

```
g = int(input('Enter a number : '))
```

```
class A:
```

```
    def __init__(self):
```

```
        # Generating a random private number selected by alice
```

```
        self.n = random.randint(1, p)
```

```
    def publish(self):
```

```
        # generating public values
```

```
        return (g**self.n)%p
```

```
    def compute_secret(self, gb):
```

```
        # computing secret key
```

```
        return (gb**self.n)%p
```

```
class B:
```

```
    def __init__(self):
```

```
        # Generating a random private number selected for alice
```

```
        self.a = random.randint(1, p)
```

```
        # Generating a random private number selected for bob
```

```
        self.b = random.randint(1, p)
```

```
self.arr = [self.a,self.b]
```

```
def publish(self, i):
```

```
    # generating public values
```

```
    return (g**self.arr[i])%p
```

```
def compute_secret(self, ga, i):
```

```
    # computing secret key
```

```
    return (ga**self.arr[i])%p
```

```
alice = A()
```

```
bob = A()
```

```
eve = B()
```

```
# Printing out the private selected number by Alice and Bob
```

```
print(f'Alice selected (a) : {alice.n}')
```

```
print(f'Bob selected (b) : {bob.n}')
```

```
print(f'Eve selected private number for Alice (c) : {eve.a}')
```

```
print(f'Eve selected private number for Bob (d) : {eve.b}')
```

```
# Generating public values
```

```
ga = alice.publish()
```

```
gb = bob.publish()
```

```
gea = eve.publish(0)
```

```
geb = eve.publish(1)
```

```
print(f'Alice published (ga): {ga}')
```

```
print(f'Bob published (gb): {gb}')
```

```
print(f'Eve published value for Alice (gc): {gea}')
```

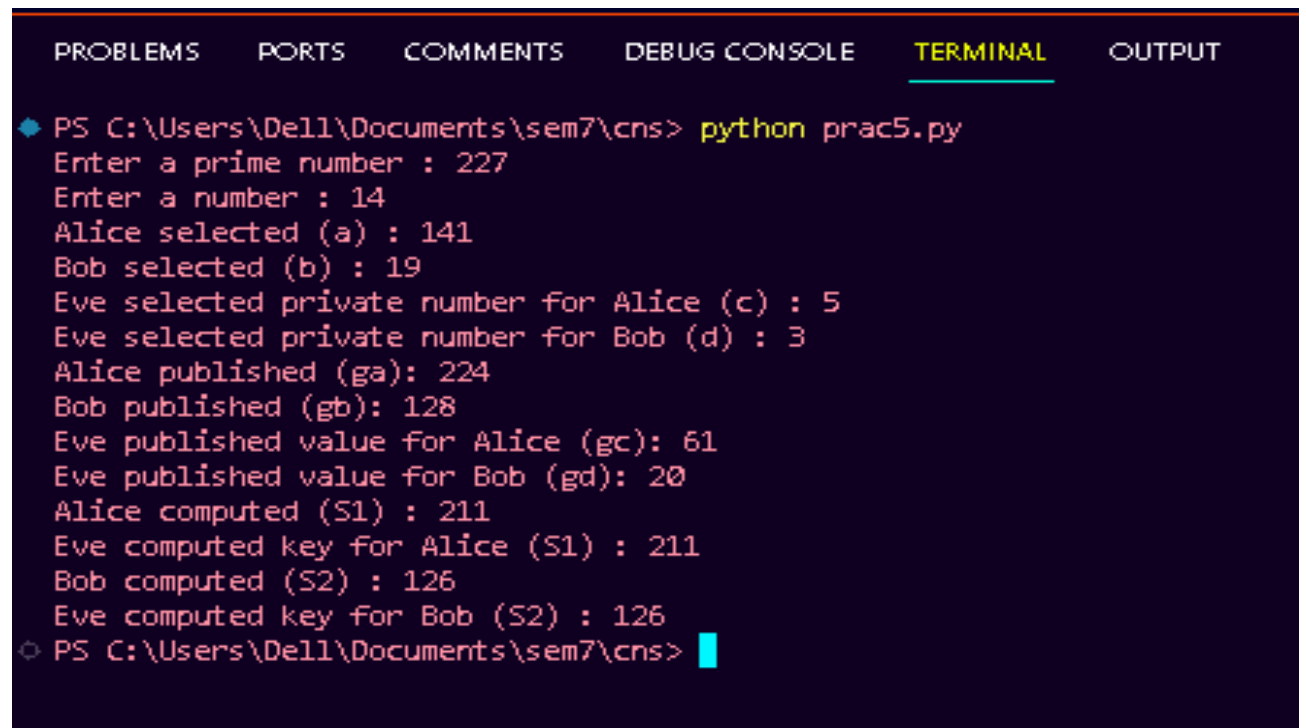
```
print(f'Eve published value for Bob (gd): {geb}')
```

```
# Computing the secret key
```



```
sa = alice.compute_secret(gea)
sea = eve.compute_secret(ga,0)
sb = bob.compute_secret(geb)
seb = eve.compute_secret(gb,1)
print(f'Alice computed (S1) : {sa}')
print(f'Eve computed key for Alice (S1) : {sea}')
print(f'Bob computed (S2) : {sb}')
print(f'Eve computed key for Bob (S2) : {seb}')
```

OUTPUT:



```
PROBLEMS  PORTS  COMMENTS  DEBUG CONSOLE  TERMINAL  OUTPUT

PS C:\Users\Dell\Documents\sem7\cns> python prac5.py
Enter a prime number : 227
Enter a number : 14
Alice selected (a) : 141
Bob selected (b) : 19
Eve selected private number for Alice (c) : 5
Eve selected private number for Bob (d) : 3
Alice published (ga): 224
Bob published (gb): 128
Eve published value for Alice (gc): 61
Eve published value for Bob (gd): 20
Alice computed (S1) : 211
Eve computed key for Alice (S1) : 211
Bob computed (S2) : 126
Eve computed key for Bob (S2) : 126
PS C:\Users\Dell\Documents\sem7\cns>
```


Code:

```
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class Pract7 {
    public static String encryptThisString(String input)
    {
        try {
            // getInstance() method is called with algorithm SHA-1
            MessageDigest md = MessageDigest.getInstance("SHA-1");

            // digest() method is called
            // to calculate message digest of the input string
            // returned as array of byte
            byte[] messageDigest = md.digest(input.getBytes());

            // Convert byte array into signum representation
            BigInteger no = new BigInteger(1, messageDigest);

            // Convert message digest into hex value
            String hashtext = no.toString(16);

            // Add preceding 0s to make it 32 bit
            while (hashtext.length() < 32) {
                hashtext = "0" + hashtext;
            }

            // return the HashText
            return hashtext;
        }
    }
}
```

```

    }

    // For specifying wrong message digest algorithms
    catch (NoSuchAlgorithmException e) {
        throw new RuntimeException(e);
    }
}

// Driver code
public static void main(String args[]) throws
                        NoSuchAlgorithmException
{

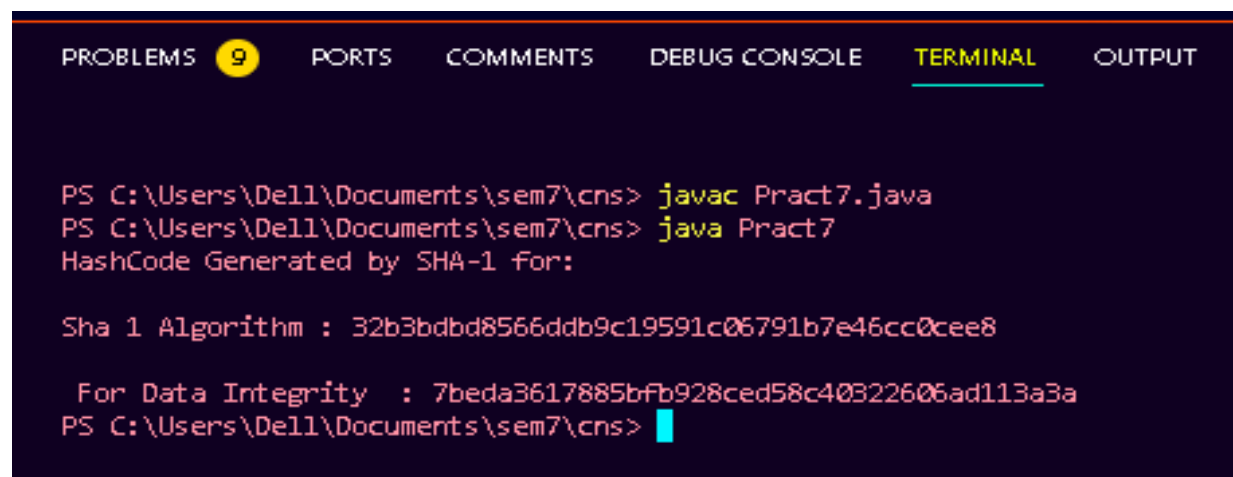
    System.out.println("HashCode Generated by SHA-1 for: ");

    String s1 = "Sha 1 Algorithm";
    System.out.println("\n" + s1 + " : " + encryptThisString(s1));

    String s2 = " For Data Integrity ";
    System.out.println("\n" + s2 + " : " + encryptThisString(s2));
}
}

```

OUTPUT:



```

PROBLEMS 9 PORTS COMMENTS DEBUG CONSOLE TERMINAL OUTPUT

PS C:\Users\Dell\Documents\sem7\cns> javac Pract7.java
PS C:\Users\Dell\Documents\sem7\cns> java Pract7
HashCode Generated by SHA-1 for:

Sha 1 Algorithm : 32b3bdbd8566ddb9c19591c06791b7e46cc0cee8

 For Data Integrity : 7beda3617885bf928ced58c40322606ad113a3a
PS C:\Users\Dell\Documents\sem7\cns>

```

Practical 6

Aim: Study different approaches for Anti-virus software and write one document.

- a. Examine files to look for viruses by means of a virus dictionary
- b. Identifying the suspicious behavior from any computer program which might indicate infection

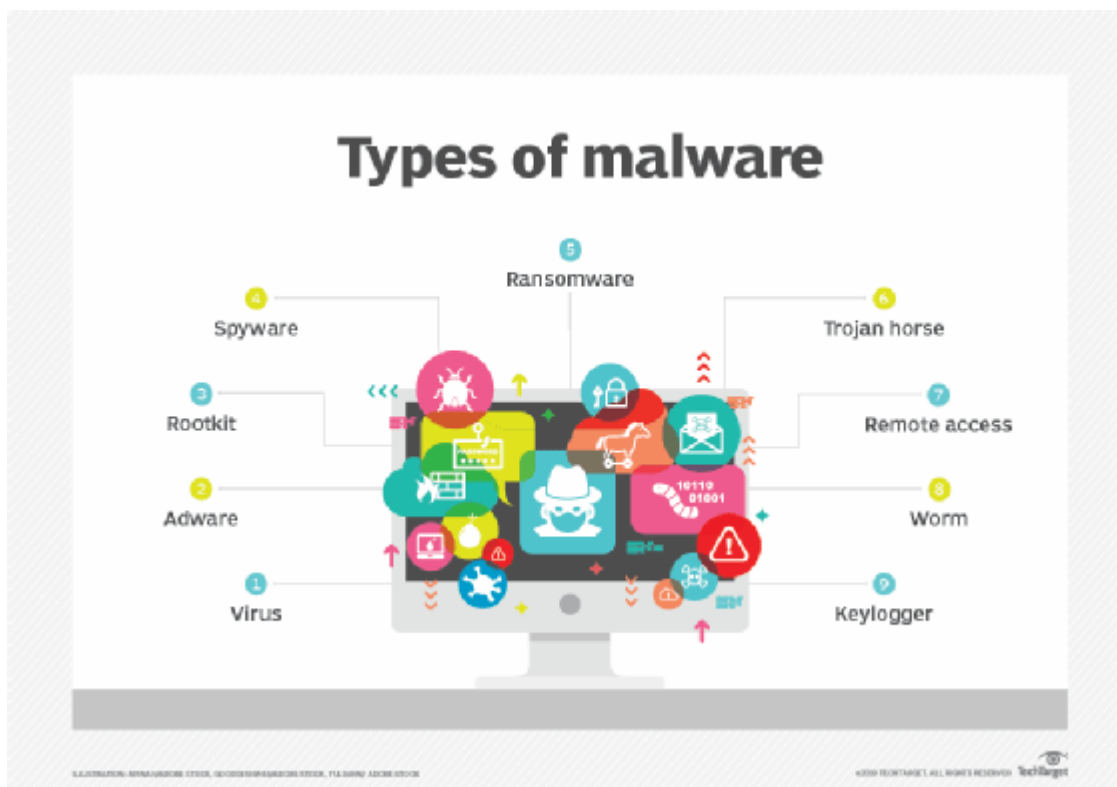
Introduction:

What is antivirus software (antivirus program)?

Antivirus software (antivirus program) is a security program designed to prevent, detect, search and remove viruses and other types of malware from computers, networks and other devices. Often included as part of a security package, antivirus software can also be purchased as a standalone option.

Typically installed on a computer as a proactive approach to cybersecurity, an antivirus program can help mitigate a variety of cyber threats, including keyloggers, browser hijackers, Trojan horses, worms, rootkits, spyware, adware, botnets, phishing attempts and ransomware attacks.

Due to the constantly evolving nature of cybercrimes and new versions of malware being released daily, including zero-day attacks, no antivirus program can offer detection and protection against all threat vectors.



virus is just one of the many types of malware that antivirus software is designed to prevent, detect, search and remove.

How antivirus software works

Antivirus software typically runs as a background process, scanning computers, servers or mobile devices to detect and restrict the spread of malware. Many antivirus software programs include real-

time threat detection and protection to guard against potential vulnerabilities and perform system scans that monitor device and system files, looking for possible risks.

Antivirus software usually performs the following basic functions: Scans directories or specific files against a library of known malicious [signatures](#) to detect abnormal patterns indicating the presence of malicious software.

3. Computer Virus Pattern

Computer virus analysis has some common patterns that lend efficiency to the analysis process, in order to stay far from the anti-virus scanners, computer viruses gradually through patterns improve their code to make them invisible. Simply put, computer virus patterns also referred to as virus signatures for those known by antiviruses are means through which viruses replicate themselves over and over as they infect computer systems. Virus signature is the representative byte-pattern part of virus family, which when a virus scanner recognizes it in a file, it notifies the user that the file is infected. A virus signature is the fingerprint of a virus. It is a set of unique data, or bits of code, that allow it to be identified. A variety of viruses may have the same virus signature allowing anti-virus programs to detect multiple viruses when looking for a single virus signature. Because of this sharing of the same virus signature between multiple viruses, antivirus programs can sometimes detect a virus that is not even known yet. Typically new viruses have a virus signature that is not used by other viruses, but new "strains" of known virus sometimes use the same virus signature as earlier strains. Computer virus authors and antivirus vendors have constantly fought in an evasion of detection game through creation of new virus signatures. Computer malwares have become more and more sophisticated, using advanced code obfuscation techniques to resist antivirus detection. Polymorphic and metamorphic computer viruses are currently the hardest kinds of viruses to detect. Both types of viruses are able to mutate into an infinite number of functionally equivalent copies of themselves. This sophistication comes with the creation of new virus patterns that are not easily detectable by the antiviruses available in the market today. Heuristic detection is a scanning mechanism that antivirus software employs in detecting for virus signatures. The heuristic detection methods encompass more than 250,000 new virus signatures and are most effective for locating new virus signatures. When there are new signatures created each time a new virus comes out these then should be detected during the virus scans since it is necessary to create the new signatures as the new viruses cannot otherwise be detected. Metamorphic type of viruses modifies their code to produce an equivalent one during their propagation. These viruses attempt to evade detection through static analysis by implementing code obfuscation techniques. A technique implemented by swapping interchangeable instructions, inserting garbage instructions and introducing conditional jumps to produce the child virus. Here the signature of a virus is broken by changing the order of instructions without altering the control flow. A sophisticated type of this virus will generate code based on the host's operating system by translating the instructions to the corresponding machine code. The detection of these viruses using their signature is challenging since the signature is broken in each version of the virus. In order to detect such metamorphic viruses, the detection system should be designed to extract the essential instructions of the virus from virus instance. This extracted instruction set should be used to detect the viruses of that type.

4. Need of Anti-virus:

- Antivirus plays an important role in any device. Following are a few of the top protection provided by it.
- Antivirus detects malware and viruses that prominently penetrate company systems. This will help in maintaining the hygiene of the computer.
- The growth of hackers is immense these days. Hackers tend to target sensitive and confidential data from your system. In the presence of antivirus, the activity of data access and more can be detected. Moreover, the data will be protected to a great extent by the antivirus software.
- If you regularly plug in external devices to your PC, the antivirus software will protect your PC/laptop from any potential virus from any external device.

- Through the above-mentioned methods, your PC/ laptop will be protected against viruses to a great extent. As a result, your computer will have a long life.

5. Benefits of Anti-virus:

5.1 Virus Protection:

The main role of an antivirus program is to face viruses and other sorts of malware. The viruses won't only cause damages to your data; it can degrade the general system performance. All of them can happen without your knowledge. The antivirus programming introduced on your PC distinguishes and eliminates this malware before they create any damages to your PC.

5.2 Spyware Protection:

Spyware because the name suggests may be quite a malware that spies on your computer stealing all the confidential information. These details also include MasterCard details, passwords and other financial data. This ultimately results in fraud. The antivirus software has the potential to stop these sorts of spyware attacks.

5.3 Web Protection:

While surfing the web, users can encounter various other sorts of threats. In untrustworthy sites, cyber attackers can gather your MasterCard and checking account details. One among the thanks for overcoming this is often by using antivirus software. Using an antivirus program you'll protect your valuable pieces of information while surfing online.

5.4 Spam Protection:

Viruses also can enter your computer through means of spam emails and ads. These emails and ads can show up repeatedly albeit you haven't any interest in it. Once the virus finds thanks to sneak into your PC it causes irreversible damages. An Antivirus works by the way of blocking these spam emails and ads.

5.5 Firewall Feature:

The firewall provides two-way protection. This suggests that regardless of the information that's sent or received is going to be double-checked here. Hence, hackers cannot enter the system data.

5.6 Cost-Effective:

Even though there are many premium versions of antivirus programs for a monthly/yearly subscription fee, there are some antivirus programs that are completely free from charge. These sorts of antivirus programs offer almost an equivalent level of protection provided by the subscription-based. Albeit you select to afford a premium version, they're relatively inexpensive.

6. Drawbacks of Antivirus:

6.1 System Slowdown:

Using an antivirus program means tons of resources from the memory and therefore the disk drive is getting used. As a result, it can drastically slow down the overall speed of the pc. Moreover, the method of scanning also can cause lags within the network.

6.2 No Complete Protection

If you're employing a free antivirus program, there's no guarantee that it'll provide you the entire protection. Moreover, they're capable of identifying only certain sorts of threats. So as for acquiring a complete level of protection, you've got to use a firewall also.

6.3 Security Holes

When security holes are present inside the OS or the networking software, it'll provide an opportunity for the virus to bypass the antivirus software. Unless the user takes action to stay updated, the antivirus software won't be effective.

6.4 Limited Detection Techniques:

For identifying a possible threat, there is always quite one method available. However, within the case of antivirus programs, it mostly executes the tactic of virus scanning. Sometimes the antivirus programs can offer you false alarms if the scanning matches with the traditional file.

6.5 Frequent Advertisements

Apart from premium versions of antivirus programs, through some means, the free antivirus software must generate an income. Advertising is one of the ways to realize them. Many sometimes these advertisements degrade the user experience.

6.6 No Customer Support

Unless you buy the premium version, there won't be any customer support given to you. Within the event of any problem, the sole thanks to overcoming are through forums and knowledge bases.

7. Identifying the suspicious behavior from any computer program which might indicate infection

A. Unexpected pop-up windows

Unexpected or unusual dialog boxes and windows can be a bad sign. Fake virus warnings claim you have security threats on your computer and usually prompt you to click a link or call a number. "One of the things we always tell people is that, as of right now, there's no way a website can tell you if your computer is infected," Armstrong said. "Sometimes, Skype will pop up a message saying, 'Urgent security vulnerability.' But Skype can't tell if your computer is infected." Legitimate protection software, such as Windows Defender and virus-scanning programs, will never prompt you to call a customer service number.

B. Random sounds

Infected computers are often programmed to respond with an audio signal to things you can't control. "They'll be things like warning beeps," Armstrong said. "When an error message pops up, a lot of times, it comes along with a warning message. Certain pieces of malware stifle that window so you can't see it. But you might still hear the warning message - a sound in the background that you didn't initiate." If you regularly hear chimes and bells from your computer that seem phantom, your computer may have a virus or malware infection.

- **MacOS antivirus software.** Although Apple macOS viruses exist, they're less common than Windows viruses, so antivirus products for Mac-based devices are less standardized than those for Windows. There are several free and paid products available, providing on-demand tools to protect against potential malware threats through full-system malware scans and the ability to sift through specific email threads, attachments and various web activities.
 - **Android antivirus software.** Android is the world's most popular mobile OS and is installed on more mobile devices than any other OS. Because most mobile malware targets Android, experts recommend all Android device users install antivirus software on their devices. Vendors offer a variety of basic free and paid premium versions of their Android antivirus software, including antitheft and remote-locating features. Some run automatic scans and actively try to stop malicious webpages and files from being opened or downloaded. Play Protect is Google's built-in malware protection for Android, which was first released with Android 8.0 Oreo, and now comes with every Android device that has Google Play services version 11 or newer installed on it.
- Virus detection techniques

Antivirus software uses a variety of virus detection techniques.

The following are six common types:

1. **Signature-based detection.** Antivirus programs typically depend on stored virus signatures -- unique strings of data that are characteristic of known malware to flag malicious software. The antivirus software uses these signatures to identify viruses it encounters that security experts have already identified and analyzed.
2. **Heuristic-based detection.** This type of detection uses an algorithm to compare the signatures of known viruses against potential threats. With heuristic-based detection, antivirus software can detect viruses that haven't been discovered yet, as well as existing viruses that have been disguised or modified and released as new viruses. However, this method can also generate false-positive matches when antivirus software detects a program behaving similarly to a malicious program and incorrectly identifies it as a virus.
3. **Behavior-based detection.** Antivirus software can also use behavior-based detection to analyze an object's behavior or potential behavior for suspicious activities and infers malicious intent based on those observations. For example, code that attempts to perform unauthorized or abnormal actions would indicate the object is malicious or, at least, suspicious. Some examples of behaviors that potentially signal danger include modifying or deleting large numbers of files, monitoring keystrokes, changing settings of other programs and remotely connecting to computers.
4. **Cloud analysis.** According to Atlas VPN, on average, hackers produced more than 316,000 malware threats daily in 2022. Since it's impossible for any antivirus program to combat the vast number of rapidly appearing malware variants, antivirus companies now provide cloud analysis as part of their antivirus offerings. Cloud analysis is a modern way of performing malware analysis, as it's done on the cloud using the antivirus vendor's servers. This way, if a malicious file or program is detected by the antivirus program, it's sent to the vendor's labs, where it's tested. If it's confirmed to be malicious, a signature is created for it, which blocks it from all the other devices where it's detected.
5. **Sandbox analysis.** This detection technique runs a program or file in a virtual sandbox environment to analyze its behavior before permitting it into the system. Using this technique, antivirus software only permits a file to execute in the real environment if the sandbox analysis confirms it to be safe. This feature is also used for running files that the antivirus program is unable to allowlist or denylist. Since the files are executed in an isolated environment, even if they end up being malicious, no harm is done to the system, as they're only executed in a virtual sandbox container.
6. **Host intrusion prevention system (HIPS).** Security and antivirus software commonly uses this technology to detect potentially malicious activities in a program using signature-based detection. A HIPS continuously monitors each activity and instantly notifies users by presenting them with authorization options, such as Allow and Block.

Challenges facing antivirus software

According to CyberCrime Magazine, 90% of the world's population, ages 6 and older, will be connected to the internet by 2030. This exponential growth in internet connections is also responsible for the significant rise in viruses and cyber attacks.

While antivirus programs were originally developed to combat viruses and cyber threats, they do come with a few limitations.

The following highlights the current and future challenges of antivirus software:

- Antivirus software that uses only signature-based detection can't expose new types of malware, including variants of existing malware. Signature-based detection can only detect new viruses when the definition file is updated with information about the new virus. With the number of new malware signatures increasing rapidly, making antimalware software based solely on signatures is impractical. However, signature-based detection doesn't usually produce false-positive matches.

- Even the best antivirus software can sometimes erroneously identify a secure piece of a program or file as malware, which can lead to a legitimate and important file or program getting quarantined or deleted by the antivirus. Free antivirus options are typically more prone to false positives than paid services, as they don't often provide enterprise-level scanning and detection of attacks and threat vectors.
- Antivirus software can sometimes interfere with system updates by either preventing them from happening or halting them in the middle. In most cases, the user must take the extra step of disabling a firewall before attempting to install system updates or firmware upgrades.
- Antivirus software runs quietly in the background and is barely noticeable, but it can consume a lot of system resources, including memory and disk space, causing a device's performance to slow down. The antivirus scanning feature can also cause noticeable lags within the network.
- Regular antivirus software provides just one layer of virus protection. For comprehensive protection, most organizations must invest in a multilayered approach, such as both hardware- and software-based firewalls or a complete internet security suite that includes antivirus options.

8. Conclusion: Anti-virus protection is, or should be, an integral part of any Information Systems operation, be it personal or professional. There are number of computer virus are created and these computer virus are affected in day today life. The large number of Anti-virus software available in the market and some are being launched, each one of them offers new features for detecting and eradicating viruses and malware. The antivirus software objective is to behind the viral protection programs is to secure the system using these 3 tasks; Take preventive measure, Detection of the malicious code, Eradication. There are six different brands of Anti-virus software that are being used by the various categories of users in the selected area. Anti-virus software performs frequent virus signature, or definition, updates. These updates are necessary for the software to detect and remove new viruses. New viruses are being created and released almost daily, which forces anti-virus software to need frequent updates

9. References [1] Usages of Selected Antivirus Software in Different Categories of Users in selected Districts March 2014 Dr. Bhaskar Vijayrao Patil, Milind Joshi. [2] Review of Viruses and Antivirus Patterns Jan 2017 Muchelule Yusuf, Waniale & Nevole, Misiko Jacob

Practical No. 8

Aim : Study and demonstrate system hacking and write a report.

- a. How to crack a password?
- b. How to use Ophcrack to Crack Passwords

Theory :

What is password cracking?

Password cracking is the process of using an application program to identify an unknown or forgotten password to a computer or network resource. It can also be used to help a threat actor obtain unauthorized access to resources. With the information malicious actors gain using password cracking, they can undertake a range of criminal activities. Those include stealing banking credentials or using the information for identity theft and fraud. A password cracker recovers passwords using various techniques. The process can involve comparing a list of words to guess passwords or the use of an algorithm to repeatedly guess the password.

What does a password cracking attack look like?

The general process a password cracker follows involves these four steps:

Steal a password via some nefarious means. That password has likely been encrypted before being stored using a hash. Hashes are mathematical functions that change arbitrary-length inputs into an encrypted fixed-length output. Choose a cracking methodology, such as a brute-force or dictionary attack, and select a cracking tool.

Prepare the password hashes for the cracking program. This is done by providing an input to the hash function to create a hash that can be authenticated.

Run the cracking tool. A password cracker may also be able to identify encrypted passwords. After retrieving the password from the computer's memory, the program may be able to decrypt it. Or, by using the same algorithm as the system program, the password cracker creates an encrypted version of the password that matches the original.

What are password cracking techniques?

Password crackers use two primary methods to identify correct passwords: brute-force and dictionary attacks. However, there are plenty of other password cracking methods, including the following:

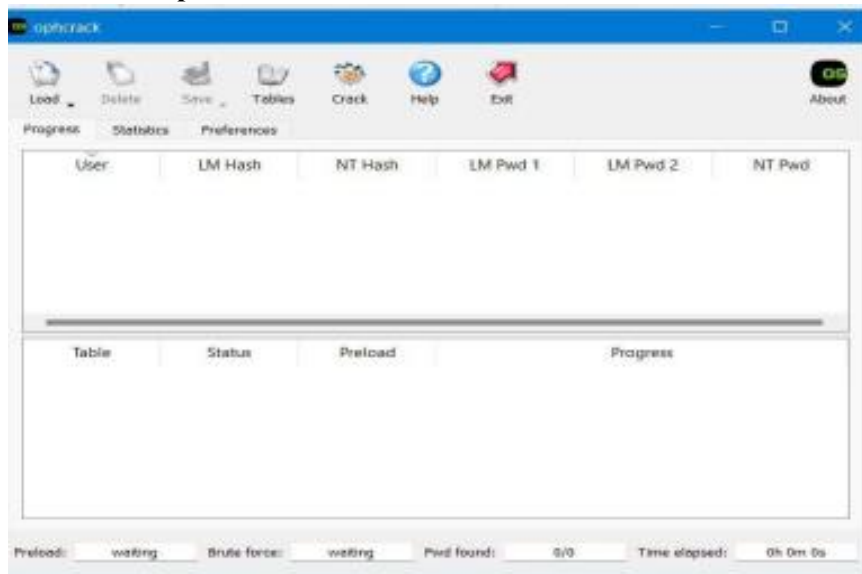
Brute force. This attack runs through combinations of characters of a predetermined length until it finds the combination that matches the password. **Dictionary search.** Here, a password cracker searches each word in the dictionary for the correct password. Password dictionaries exist for a variety of topics and combinations of topics, including politics, movies and music groups.

Phishing. These attacks are used to gain access to user passwords without the use of a password cracking tool. Instead, a user is fooled into clicking on an email attachment. From here, the attachment could install malware or prompt the user to use their email to sign into a false version of a website, revealing their password.

Malware. Similar to phishing, using malware is another method of gaining unauthored access to passwords without the use of a password cracking tool. Malware such as keyloggers, which track keystrokes, or screen scrapers, which take screenshots, are used instead.

Guessing. An attacker may be able to guess a password without the use of tools. If the threat actor has enough information about the victim or the victim is using a common enough password, they may be able to come up with the correct characters. Some password cracking programs may use hybrid attack methodologies where they search for combinations of dictionary entries and numbers or special characters. For example, a password cracker may search for ants01, ants02, ants03, etc. This can be helpful when users have been advised to include a number in their password.

How to use Ophcrack to Crack Passwords?



Ophcrack is a free, open-source tool that can be used to recover lost Windows passwords. It works by using pre-computed tables to crack password hashes, allowing users to recover their forgotten passwords quickly and easily. In this article, we will take a look at how to use Ophcrack for Windows password recovery, with step-by-step instructions and examples. Before we begin, it's important to note that Ophcrack is only able to recover passwords for local Windows accounts, and not for Microsoft accounts. If you are using a Microsoft account to sign in to your Windows computer, you will need to reset your password through the Microsoft account website.

With that said, let's take a look at how to use Ophcrack for Windows password recovery.

Step 1: Download and Install Ophcrack

The first step in using Ophcrack for Windows password recovery is to download and install the tool. You can download the latest version of Ophcrack from the official website at <https://ophcrack.github.io/>. Once the download is complete, run the installer and follow the prompts to install Ophcrack on your computer.

Step 2: Create a Bootable Ophcrack USB or CD

Next, you will need to create a bootable Ophcrack USB or CD. This will allow you to boot your computer from the Ophcrack USB or CD, allowing you to access the Ophcrack software and recover your lost password. To create a bootable Ophcrack USB, you will need a USB drive with at least 1 GB of storage space and a tool such as Rufus to create the bootable USB. To create a bootable Ophcrack CD, you will need a blank CD and a tool such as ImgBurn to create the bootable CD. Once you have your bootable Ophcrack USB or CD ready, move on to the next step.

Step 3: Boot Your Computer from the Ophcrack USB or CD

With your bootable Ophcrack USB or CD ready, it's time to boot your computer from it. To do this, you will need to enter your computer's BIOS or UEFI settings and change the boot order. The exact steps for entering the BIOS or UEFI settings and changing the boot order will vary depending on your computer's make and model. In general, you will need to press a key (such as F2 or Del) during the boot process to enter the BIOS or UEFI settings, and then navigate to the "Boot" or "Boot Order" settings and change the order so that the Ophcrack USB or CD is first in the list. Once you have changed the boot order, save your changes and exit the BIOS or UEFI settings. Your computer should now boot from the Ophcrack USB or CD.

Step 4: Use Ophcrack to Recover Your Lost Password

With your computer booted from the Ophcrack USB or CD, you can now use the Ophcrack software to recover your lost password. Upon booting, Ophcrack will automatically detect all of the user accounts on your computer and display them in a list. Simply select the user account for which you want to recover the password, and Ophcrack will begin the cracking process. Depending on the complexity of the password, the cracking process may take some time. Ophcrack will use the pre-computed tables to try different password combinations and crack the password hash. Once the password has been recovered, it will be displayed on the screen.

Conclusion : In this practical we studied how the passwords are cracked using system hacking and how to use Ophcrack to crack the password of windows system