DAYANANDA SAGAR COLLEGE OF ENGINEERING

(An Autonomous Institute affiliated to Visvesvaraya Technological University (VTU), Belagavi,
Approved by AICTE and UGC, Accredited by NAAC with 'A' grade & ISO 9001-2015 Certified Institution)

Shavige Malleshwara Hills, Kumaraswamy Layout, Bengaluru - 560 111. India

Department of Computer Science and Business Systems

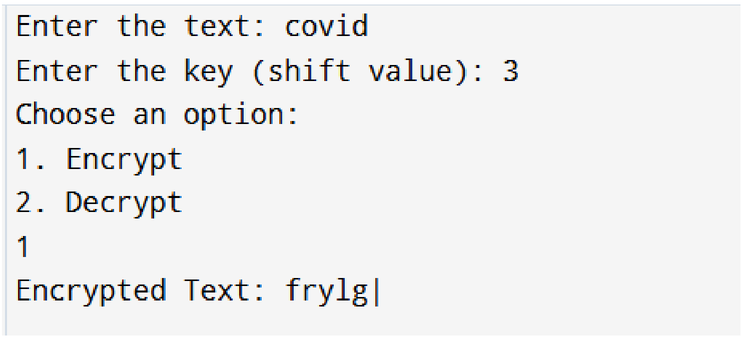
Information Security Lab Manual

Sub Code:22CB63 **Sem:6th**

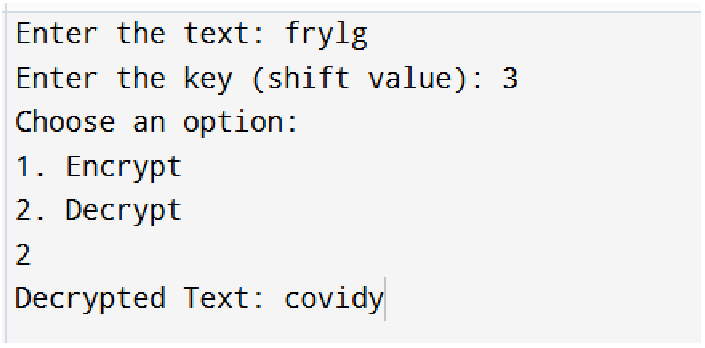
Sl No	Experiment Name
1	Perform encryption, decryption using the following substitution techniques (i). Ceaser cipher, (ii) Playfair cipher
2	Implement the Data Encryption Standard (DES) Algorithm (User Message Encryption).
3	Implement the Diffie-Hellman Key Exchange algorithm for a given problem.
4	Calculate the message digest of a text using the SHA-256 algorithm.
5	Demonstrate intrusion detection system (IDS)
6	Demonstrate the Administration of users, password policies, privileges and roles.
7	Implementation of discretionary access control and mandatory access control.
8	Working with Sniffers for monitoring network communication.

1. **Perform encryption, decryption using the following substitution techniques**
(i). Ceaser cipher, (ii) Playfair cipher

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>

// Function to encrypt the text using Caesar Cipher
void encryptCaesar(char text[], int shift) {
    for (int i = 0; text[i] != '\0'; i++) {
        char ch = text[i];
        if (isalpha(ch)) {
            char base = isupper(ch) ? 'A' : 'a';
            text[i] = (ch - base + shift) % 26 + base;
        }
    }
    printf("Encrypted Text: %s\n", text);
}

// Function to decrypt the text using Caesar Cipher
void decryptCaesar(char text[], int shift) {
    encryptCaesar(text, 26 - shift);  // Decrypt by reversing the shift
}

int main() {
    char text[100];
    int shift;

    // Input text and shift value
    printf("Enter the text: ");
    fgets(text, sizeof(text), stdin);
    text[strcspn(text, "\n")] = '\0';  // Remove newline character if present

    printf("Enter the shift value: ");
    scanf("%d", &shift);

    // Encryption
    char encryptedText[100];
    strcpy(encryptedText, text);
    encryptCaesar(encryptedText, shift);

    // Decryption
    char decryptedText[100];
    strcpy(decryptedText, encryptedText);
    decryptCaesar(decryptedText, shift);

    printf("Decrypted Text: %s\n", decryptedText);

    return 0;
}
```



```
Enter the text: covid
Enter the key (shift value): 3
Choose an option:
1. Encrypt
2. Decrypt
1
Encrypted Text: frylg
```



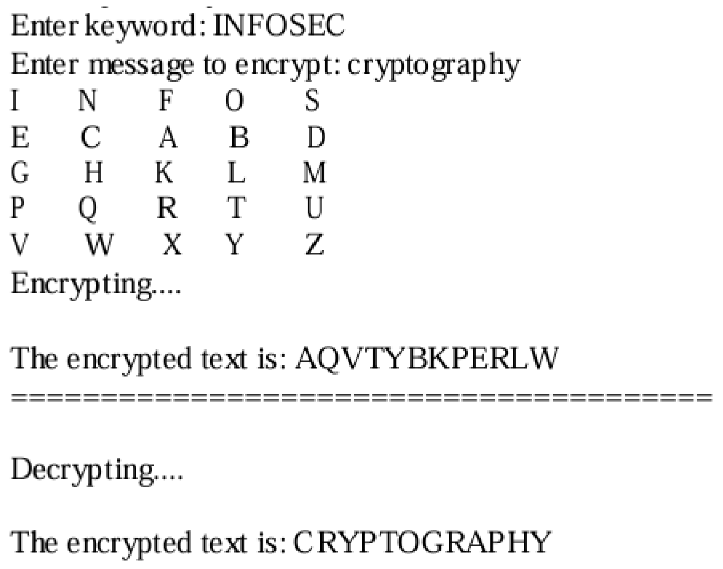
```
Enter the text: frylg
Enter the key (shift value): 3
Choose an option:
1. Encrypt
2. Decrypt
2
Decrypted Text: covidy
```

ii. Playfair cipher

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define SIZE 5  // Size of the playfair matrix

char matrix [SIZE][SIZE];

// Function to remove duplicate characters from the key and build the Playfair matrix
void create Matrix(char key[]) {
   int alphabet [26] = {0};
   int row = 0, col = 0;

   for (int i = 0; i < strlen(key); i++) {
      if (isalpha(key[i])) {
         char ch = tolower(key[i]) == 'j' ? 'i' : tolower(key[i]);
         if (alphabet[ch - 'a'] == 0) {
            matrix[row][col++] = ch;
            alphabet[ch - 'a'] = 1;
            if (col == SIZE) {
               col = 0;
               row++;
            }
         }
      }
   }
   // Fill in the remaining letters of the alphabet
```

```
    for (char ch = 'a'; ch <= 'z'; ch++) {
        if (ch == 'j') continue;  // Skip 'j'
        if (alphabet[ch - 'a'] == 0) {
            matrix[row][col++] = ch;
            alphabet[ch - 'a'] = 1;
            if (col == SIZE) {
                col = 0;
                row++;
            }
        }
    }
}
// Function to display the Playfair matrix
void displayMatrix() {
    printf("Playfair Matrix:\n");
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            printf("%c ", matrix[i][j]);
        }
        printf("\n");
    }
}
// Function to prepare the input text by removing spaces and adjusting digraphs
void prepareText(char text[]) {
    int len = strlen(text);
    int index = 0;
    char temp[100] = {0};

    for (int i = 0; i < len; i++) {
        if (isalpha(text[i])) {
            text[index++] = tolower(text[i]);
        }
    }
    text[index] = '\0';

    // Adjust text with 'x' for repeated letters and ensure even length
    index = 0;
    len = strlen(text);
    for (int i = 0; i < len; i++) {
        temp[index++] = text[i];
        if (i < len - 1 && text[i] == text[i + 1]) {
            temp[index++] = 'x';
        }
    }
    if (index % 2 != 0) {
        temp[index++] = 'x';
    }

temp[index] = '\0';
    strcpy(text, temp);
}
```

```
// Function to find the position of a letter in the Playfair matrix
void findPosition(char ch, int *row, int *col) {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (matrix[i][j] == ch) {
                *row = i;
                *col = j;
                return;
            }
        }
    }
}

// Encryption function using Playfair cipher
void encrypt(char text[]) {
    int row1, col1, row2, col2;
    printf("Encrypted Text: ");

    for (int i = 0; i < strlen(text); i += 2) {
        findPosition(text[i], &row1, &col1);
        findPosition(text[i + 1], &row2, &col2);

        if (row1 == row2) {  // Same row
            printf("%c%c", matrix[row1][(col1 + 1) % SIZE], matrix[row2][(col2 + 1) % SIZE]);
        } else if (col1 == col2) {  // Same column
            printf("%c%c", matrix[(row1 + 1) % SIZE][col1], matrix[(row2 + 1) % SIZE][col2]);
        } else {  // Rectangle swap
            printf("%c%c", matrix[row1][col2], matrix[row2][col1]);
        }
    }
    printf("\n");
}

// Decryption function using Playfair cipher
void decrypt(char text[]) {
    int row1, col1, row2, col2;
    printf("Decrypted Text: ");

    for (int i = 0; i < strlen(text); i += 2) {
        findPosition(text[i], &row1, &col1);
        findPosition(text[i + 1], &row2, &col2);

        if (row1 == row2) {  // Same row
            printf("%c%c", matrix[row1][(col1 - 1 + SIZE) % SIZE], matrix[row2][(col2 - 1 + SIZE) %
SIZE]);
        } else if (col1 == col2) {  // Same column
            printf("%c%c", matrix[(row1 - 1 + SIZE) % SIZE][col1], matrix[(row2 - 1 + SIZE) %
SIZE][col2]);
        } else {  // Rectangle swap
            printf("%c%c", matrix[row1][col2], matrix[row2][col1]);
        }
    }
    printf("\n");
}

int main() {
    char key[100], text[100];

    printf("Enter the key: ");
    fgets(key, sizeof(key), stdin);
    key[strcspn(key, "\n")] = '\0';  // Remove the newline character

    printf("Enter the text: ");
    fgets(text, sizeof(text), stdin);
    text[strcspn(text, "\n")] = '\0';  // Remove the newline character

    createMatrix(key);
    displayMatrix();

    prepareText(text);
    printf("Prepared Text: %s\n", text);

    encrypt(text);
    decrypt(text);

    return 0;
}
```



```
Enter keyword: INFOSEC
Enter message to encrypt: cryptography
I    N    F    O    S
E    C    A    B    D
G    H    K    L    M
P    Q    R    T    U
V    W    X    Y    Z
Encrypting....

The encrypted text is: AQVTYBKPERLW
========================================

Decrypting....

The encrypted text is: CRYPTOGRAPHY
```

2. Implement the Data Encryption Standard (DES) Algorithm.

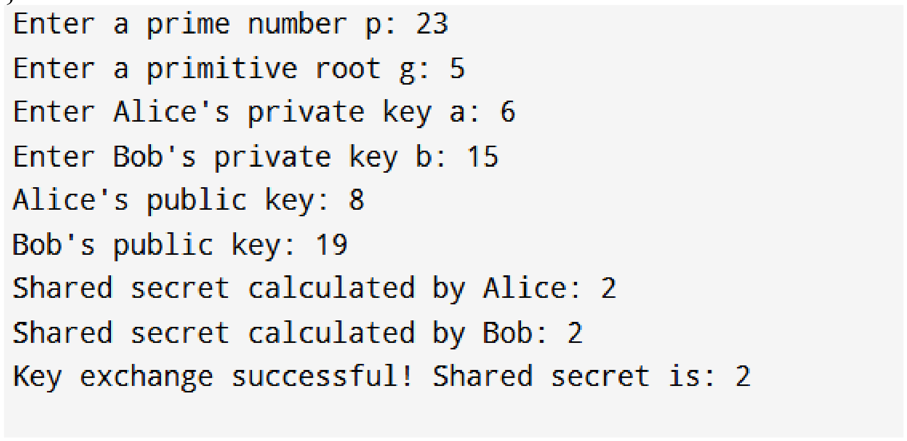
```
#include <stdio.h>
#include <stdint.h>
#include <string.h>
#define DES_BLOCK_SIZE 64
#define DES_ROUNDS 16
static const uint8_t IP[DES_BLOCK_SIZE] = {
 58, 50, 42, 34, 26, 18, 10, 2,
 60, 52, 44, 36, 28, 20, 12, 4,
 62, 54, 46, 38, 30, 22, 14, 6,
 64, 56, 48, 40, 32, 24, 16, 8,
 57, 49, 41, 33, 25, 17, 9, 1,
 59, 51, 43, 35, 27, 19, 11, 3,
 61, 53, 45, 37, 29, 21, 13, 5,
 63, 55, 47, 39, 31, 23, 15, 7
};
void feistel_function(uint64_t *half_block, uint64_t key) {
 *half_block ^= key;
}
void permute(uint64_t *block, const uint8_t *table, int size) {
 uint64_t result = 0;
 for (int i = 0; i < size; i++) {
 result |= ((*block >> (64 - table[i])) & 1) << (63 - i);
 }
 *block = result;
}
void key_schedule(uint64_t key, uint64_t round_keys[DES_ROUNDS]) {
 for (int i = 0; i < DES_ROUNDS; i++) {
 round_keys[i] = key ^ (i + 1);
 }
}
void des_round(uint64_t *block, uint64_t key) {
 uint64_t left = (*block >> 32) & 0xFFFFFFFF;
 uint64_t right = *block & 0xFFFFFFFF;
 feistel_function(&right, key);
 *block = (right << 32) | left;
}
void des_encrypt(uint64_t *block, uint64_t key) {
 uint64_t round_keys[DES_ROUNDS];
 key_schedule(key, round_keys);
 permute(block, IP, DES_BLOCK_SIZE);

 for (int i = 0; i < DES_ROUNDS; i++) {
 des_round(block, round_keys[i]);
 }
}
uint64_t string_to_uint64(const char *str) {
 uint64_t result = 0;
 memcpy(&result, str, strlen(str) > 8 ? 8 : strlen(str));
 return result;
}
int main() {
 char plaintext[9], key_str[9];
 uint64_t block, key;
 printf("Enter 8-character plaintext: ");
 scanf("%8s", plaintext);

 printf("Enter 8-character key: ");
 scanf("%8s", key_str);

 block = string_to_uint64(plaintext);
 key = string_to_uint64(key_str) & 0xFFFFFFFFFFFFFF00;
 printf("\nOriginal: 0x%016llX\n", block);
 des_encrypt(&block, key);
 printf("Encrypted: 0x%016llX\n", block);
 return 0;
}
```

OUTPUT:

```
1. Enter 8-character plaintext: hello123

Enter 8-character key: secret12

Original: 0x3332316F6C6C6568

Encrypted: 0xF807785D00FFB80B

2.Enter 8-character plaintext: CSBS1234

Enter 8-character key: SECRET24

Original: 0x3433323153425343

Encrypted: 0xF05F01CA000F00F6
```

3. **Implement the Diffie-Hellman Key Exchange algorithm for a given problem.**

```
#include <stdio.h>
#include <math.h>

// Function to perform modular exponentiation
// It returns (base^exp) % mod
long long power(long long base, long long exp, long long mod) {
    long long result = 1;
    base = base % mod;
    while (exp > 0) {
        if (exp % 2 == 1) {  // If exp is odd
            result = (result * base) % mod;
        }
        exp = exp >> 1;  // exp = exp // 2
        base = (base * base) % mod;
    }
    return result;
}

int main() {
    long long p, g, a, b, A, B, shared_secret_Alice, shared_secret_Bob;

    // Step 1: Public parameters (p and g)
    printf("Enter a prime number p: ");
    scanf("%lld", &p);
    printf("Enter a primitive root g: ");
    scanf("%lld", &g);

    // Step 2: Private keys (a and b)
    printf("Enter Alice's private key a: ");
    scanf("%lld", &a);
    printf("Enter Bob's private key b: ");
    scanf("%lld", &b);

    // Step 3: Calculate public keys (A and B)
    A = power(g, a, p); // A = g^a % p
    B = power(g, b, p); // B = g^b % p

    printf("Alice's public key: %lld\n", A);
    printf("Bob's public key: %lld\n", B);

    // Step 4: Exchange public keys

    // Step 5: Calculate shared secret (using the other's public key)
    shared_secret_Alice = power(B, a, p); // shared secret = B^a % p
    shared_secret_Bob = power(A, b, p);   // shared secret = A^b % p

    printf("Shared secret calculated by Alice: %lld\n", shared_secret_Alice);
    printf("Shared secret calculated by Bob: %lld\n", shared_secret_Bob);
    if (shared_secret_Alice == shared_secret_Bob) {
        printf("Key exchange successful! Shared secret is: %lld\n", shared_secret_Alice);
    } else {
        printf("Key exchange failed.\n");
    }

    return 0;
}
```



```
Enter a prime number p: 23
Enter a primitive root g: 5
Enter Alice's private key a: 6
Enter Bob's private key b: 15
Alice's public key: 8
Bob's public key: 19
Shared secret calculated by Alice: 2
Shared secret calculated by Bob: 2
Key exchange successful! Shared secret is: 2
```

4. **Calculate the message digest of a text using the SHA-256 algorithm**.

```
#include <stdio.h>
#include <stdint.h>
#include <string.h>
#include <math.h>
#define ROTR (x, n) ((x >> n) | (x << (32 - n)))
static uint32_t h[8] = {
 0x6a09e667, 0xbb67ae85, 0x3c6ef372, 0xa54ff53a,
 0x510e527f, 0x9b05688c, 0x1f83d9ab, 0x5be0cd19
};
void generate_constants(uint32_t k [64]) {
 int count = 0, num = 2;
 while (count < 64) {
 int is_prime = 1;
 for (int i = 2; i * i <= num; i++) {
 if (num % i == 0) {
 is_prime = 0;
 break;
 }
 }
 if (is_prime) {
 double cube_root = cbrt((double)num);
 k[count] = (uint32_t)(cube_root * (1ULL << 32));
 count++;
 }
 num++;
 }
}
void sha256_pad (const uint8_t *message, size_t length, uint8_t *padded,
size_t *padded_length) {
 *padded_length = (length + 9 + 63) & ~63;
 memcpy(padded, message, length);
 padded[length] = 0x80;
 memset(padded + length + 1, 0, *padded_length - length - 9);
 uint64_t bit_length = length * 8;
 for (int i = 0; i < 8; i++)
 padded[*padded_length - 8 + i] = (bit_length >> (56 - 8 * i)) &
0xFF;
}
void sha256_process_block (const uint8_t *block, uint32_t k[64]) {
 uint32_t w[64], a, b, c, d, e, f, g, h1;

 for (int i = 0; i < 16; i++)
 w[i] = (block[i * 4] << 24) | (block[i * 4 + 1] << 16) |
 (block[i * 4 + 2] << 8) | (block[i * 4 + 3]);
 for (int i = 16; i < 64; i++) {
 uint32_t s0 = ROTR(w[i-15], 7) ^ ROTR(w[i-15], 18) ^ (w[i-15] >>
3);
 uint32_t s1 = ROTR(w[i-2], 17) ^ ROTR(w[i-2], 19) ^ (w[i-2] >>
10);
```

```
 w[i] = w[i-16] + s0 + w[i-7] + s1;
 }
 a = h[0]; b = h[1]; c = h[2]; d = h[3];
 e = h[4]; f = h[5]; g = h[6]; h1 = h[7];
 for (int i = 0; i < 64; i++) {
 uint32_t S1 = ROTR(e, 6) ^ ROTR(e, 11) ^ ROTR(e, 25);
 uint32_t ch = (e & f) ^ (~e & g);
 uint32_t temp1 = h1 + S1 + ch + k[i] + w[i];
 uint32_t S0 = ROTR(a, 2) ^ ROTR(a, 13) ^ ROTR(a, 22);
 uint32_t maj = (a & b) ^ (a & c) ^ (b & c);
 uint32_t temp2 = S0 + maj;
 h1 = g; g = f; f = e; e = d + temp1;
 d = c; c = b; b = a; a = temp1 + temp2;
 }
 h[0] += a; h[1] += b; h[2] += c; h[3] += d;
 h[4] += e; h[5] += f; h[6] += g; h[7] += h1;
}
void sha256(const uint8_t *message, size_t length, uint8_t *digest) {
 uint8_t padded[128]; size_t padded_length;
 uint32_t k[64];
 generate_constants(k);
 sha256_pad(message, length, padded, &padded_length);
 for (size_t i = 0; i < padded_length; i += 64)
 sha256_process_block(padded + i, k);
 for (int i = 0; i < 8; i++)
 for (int j = 0; j < 4; j++)
 digest [i * 4 + j] = (h[i] >> (24 - 8 * j)) & 0xFF;
}
int main() {
 char message[256];
 uint8_t hash[32];
 printf("Enter message: ");
 fgets(message, sizeof(message), stdin);
 size_t length = strlen(message) - 1;
 sha256((uint8_t *)message, length, hash);
 printf("SHA-256 Hash: ");
 for (int i = 0; i < 32; i++)
 printf("%02x", hash[i]);
 printf("\n");
 return 0;
}
```
OUTPUT:

```
1. Enter message: hello123
SHA-256 Hash:
27cc6994fc1c01ce6659c6bddca9b69c4c6a9418065e612c69d110b3f7b11f8a
2. Enter message: CSBS
SHA-256 Hash:
a1a7f0f7bc5318eaaa589432d0adba26289642399eecbf956ac9bc968082c0e0
```

5. Demonstrate intrusion detection system (IDS).

An Intrusion Detection System (IDS) is a security tool that monitors network or system activities for malicious activities, policy violations, or security threats. It helps detect unauthorized access, malware, or suspicious behavior by analyzing network traffic, log files, or system activities.

Tool Applied: SNORT

Snort is an open-source Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) developed by Cisco. It monitors network traffic in real time to detect and prevent security threats such as malware, unauthorized access, and exploits.

How it works:

1. Packet Capturing

Snort captures packets from the network using libpcap, the same library used by Wireshark and TCPDump. It inspects each packet passing through the network interface.

2. Packet Analysis and Detection Snort processes packets in multiple phases to identify potential threats.
3. Logging & Alerting Once Snort detects a suspicious packet, it logs the event and generates an alert.

Process and Code

This Project is performed using Windows operating System 1. Inputs Required
• Network Traffic: Simulated ICMP packets (ping requests).
• Snort Configuration File: Custom rules and settings.
• Windows-Compatible Tools:

- Snort for Windows (precompiled binary).
- Npcap (packet capture driver).

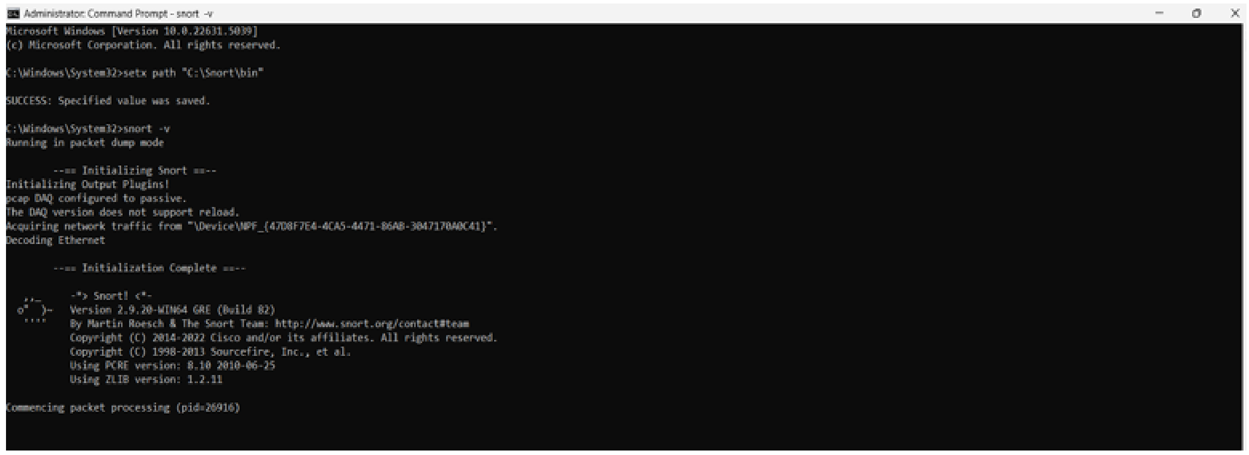
2. **Prerequisites**
   • OS: Windows 10/11.
   • Snort Installed: Downloaded and configured for Windows.
   • Npcap: Required for packet capture (replaces WinPcap).
   • Administrator Access: To install drivers and run Snort.

3. **Installation & Setup**

   Step 1: Install Npcap
   Download and install Npcap from https://npcap.com/
   Select "Install in WinPcap API-compatible Mode" during setup. https://npcap.com.

   Step 2: Install Snort for Windows 1. Download the Windows installer from Snort.org. Choose the Latest Stable Release (e.g., Snort_2_9_20_Installer.exe). 2. Run the installer and select a directory (e.g., C:\Snort). Ensure the "Install Rules" option is checked.

Step 3: Verify Installation Open Command Prompt (as Administrator) and run:

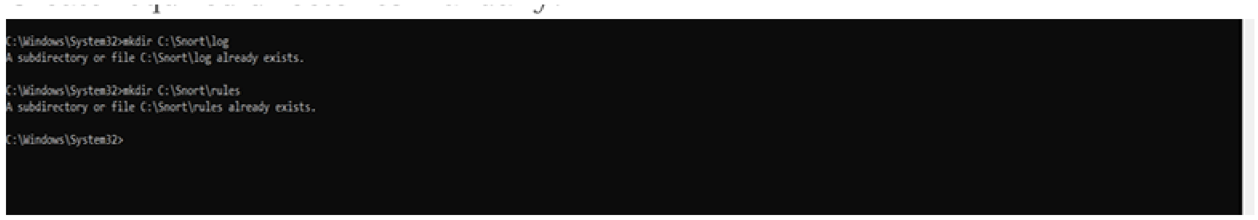


Administrator: Command Prompt - snort -v

```
Microsoft Windows [Version 10.0.22631.5039]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>setx path "C:\Snort\bin"

SUCCESS: Specified value was saved.

C:\Windows\System32>snort -v
Running in packet dump mode

        --== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
The DAQ version does not support reload.
Acquiring network traffic from "\Device\NPF_{47D8F7E4-4CA5-4471-86AB-3047170A0C41}".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.20-WIN64 GRE (Build 82)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2022 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.11

Commencing packet processing (pid=26916)
```

4. **Configuration**

Step 1: Set Up Snort Directories Create required directories manually:



```
C:\Windows\System32>mkdir C:\Snort\log
A subdirectory or file C:\Snort\log already exists.

C:\Windows\System32>mkdir C:\Snort\rules
A subdirectory or file C:\Snort\rules already exists.

C:\Windows\System32>
```

Step 2: Configure snort.conf

1. Navigate to C:\Snort\etc\snort.conf.
2. Edit the file using Notepad or a text editor like Notepad++.
3. Update Paths: Replace Linux-style paths with Windows paths:

Step 3: Create a Custom Rule

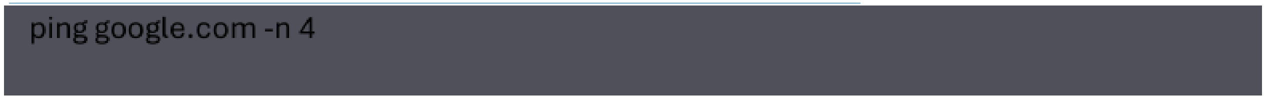
1. Navigate to the Rules Directory
    a. Go to C:\Snort\rules
    b. If custom.rules does not exist, create a new file:
        i. Open Notepad++ or Notepad
        ii. Save the file as custom.rules (ensure the file extension is .rules and not .txt)
2. Add the Open custom.rules and add: ICMP Detection Rule alert icmp any any -> any any (msg:"ICMP Ping Detected"; sid:1000001; rev
    a. alert → Triggers an alert
    b. icmp → Detects ICMP (ping) packets
    c. any any -> any any → Matches any source to any destination
    d. msg: → Message displayed when an alert is triggered
    e. sid: → Unique Snort rule ID (custom rules should be >= 1000000)
    f. rev: → Revision number of the rule
3. Save the File

5. Process

Step 1: Run Snort in IDS Mode Open Command Prompt (as Administrator) and run:
snort -i "Ethernet" -A console -q -c C:\Snort\etc\snort.conf -l C:\Snort\log.

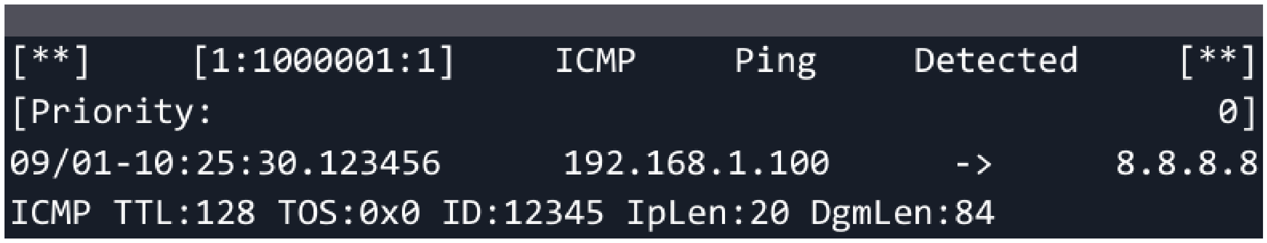
• -i "Ethernet": Replace Ethernet with your network interface name (find using snort -W).
• -l C:\Snort\log: Save logs to the specified directory.

Step 2: Generate Test Traffic
Open a new Command Prompt and ping a remote server:



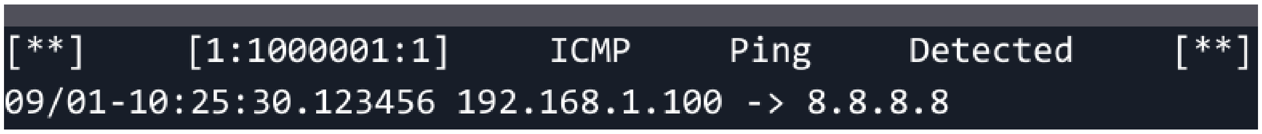
```
ping google.com -n 4
```

6. Output
Real-Time Alerts
Snort displays alerts in the console:



```
[**]     [1:1000001:1]     ICMP     Ping     Detected     [**]
[Priority:                                                  0]
09/01-10:25:30.123456      192.168.1.100      ->      8.8.8.8
ICMP TTL:128 TOS:0x0 ID:12345 IpLen:20 DgmLen:84
```

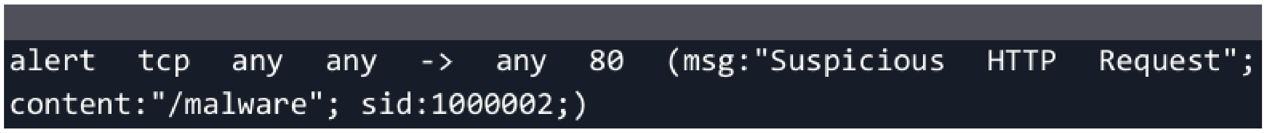
Log Files
Alerts are saved to C:\Snort\log\alert.ids:



```
[**]     [1:1000001:1]     ICMP     Ping     Detected     [**]
09/01-10:25:30.123456 192.168.1.100 -> 8.8.8.8
```

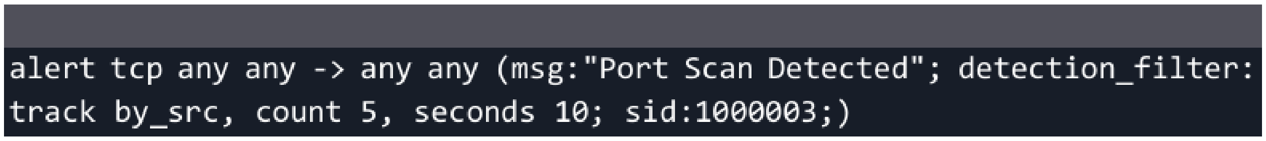
7. Explanation Key Differences from Linux
   • Interface Names: Use snort -W to list interfaces (e.g., Ethernet, Wi-Fi).
   • File Paths: Use Windows-style paths (e.g., C:\Snort\etc\snort.conf).
   • Drivers: Npcap replaces libpcap for packet capture.

Rule Customization
• Detect HTTP requests to suspicious URLs:



```
alert  tcp  any  any  ->  any  80  (msg:"Suspicious  HTTP  Request";
content:"/malware"; sid:1000002;)
```

• Block port scans:



```
alert tcp any any -> any any (msg:"Port Scan Detected"; detection_filter:
track by_src, count 5, seconds 10; sid:1000003;)
```

8. Troubleshooting Common Issues
   • No Alerts:
      o Verify the interface name with snort -W.

  - o Check custom.rules syntax and ensure it’s included in snort.conf.
  - o Disable Windows Firewall temporarily for testing.
- Permission Errors: Always run Command Prompt as Administrator.
- Stop Snort Press Ctrl+C in the Snort terminal to stop monitoring.

Conclusion

You’ve configured Snort as a Network-based IDS on Windows to detect ICMP traffic. Expand by:
• Adding rules for SQL injection, malware, or DDoS attacks.
• Integrating with Splunk or ELK Stack for log analysis. Snort on Windows provides enterprise-grade network monitoring for non-Linux environments.

6. **Demonstrate the Administration of users, password policies, privileges and roles.**
7. **Implementation of discretionary access control and mandatory access control.**

Discretionary Access Control (DAC) and Mandatory Access Control (MAC) are two fundamental access control models used in security frameworks. DAC is a flexible access control model where resource owners have the authority to grant or restrict access to their files or data. It is commonly used in operating systems like Windows and Linux, where users can set permissions (read, write, execute) on their files. It is often criticized for its vulnerability to insider threats and malware, as users can unintentionally grant access to unauthorized entities.

MAC is a more rigid and structured access control model where access permissions are determined by a central authority based on security policies. In MAC, users and resources are assigned security labels (such as "Confidential" or "Top Secret"), and access is granted or denied strictly according to these classifications. It is commonly used in military and government systems where data confidentiality is a top priority. Unlike DAC, MAC prevents users from modifying access rights, making it more secure but less flexible for general use.

Tool Used

The implementation of Discretionary Access Control (DAC) and Mandatory Access Control (MAC) can be achieved using Flask and Flask-Principal, which are Python-based tools for managing user authentication and authorization. Flask is a lightweight web framework that allows developers to build secure applications, while Flask-Principal provides a flexible permission management system based on identity and roles. In DAC implementation, Flask-Principal enables resource owners to define access controls by assigning roles and permissions dynamically. Users can set who can access, modify, or delete resources based on predefined rules. This makes DAC highly customizable but also susceptible to misconfigurations if not managed properly.

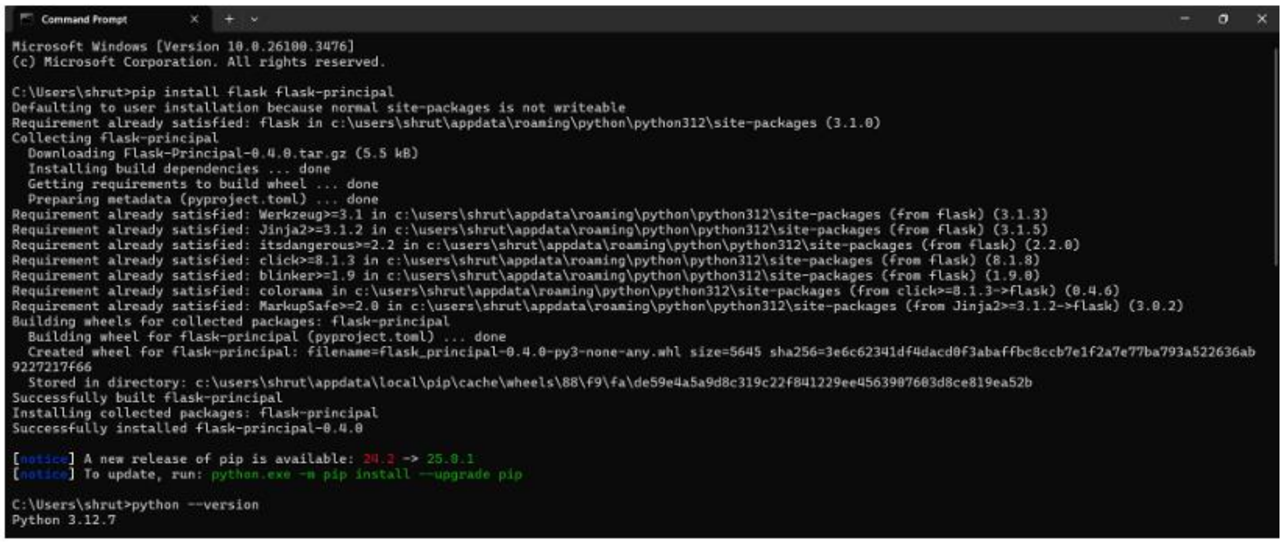
For MAC implementation, Flask-Principal enforces strict access rules by assigning security labels to users and resources, ensuring that permissions are dictated by an overarching policy. In this setup, security levels such as “Confidential” or “Top Secret” are predefined, and access decisions are enforced by the system rather than individual users. Flask’s request handling and Flask-Principal’s role-based authorization allow developers to implement these constraints programmatically. This approach enhances security by preventing unauthorized access, making it ideal for high-security applications. However, it reduces flexibility, as users cannot override security policies, ensuring strict adherence to access rules.

Procedure

Access control mechanisms like DAC (Discretionary Access Control) and MAC (Mandatory Access Control) are essential for securing applications. This guide will walk you through the process of implementing these access control models using Flask and Flask-Principal.

Step 1: Install Required Dependencies

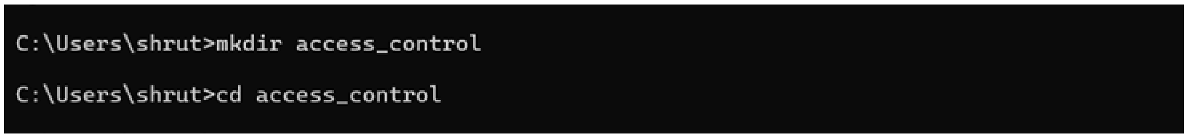
Before writing the access control logic, we need to install the required Python packages.
● Flask: A lightweight web framework for Python.
● Flask-Principal: A library for role-based access control.



```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

C:\Users\shrut>pip install flask flask-principal
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: flask in c:\users\shrut\appdata\roaming\python\python312\site-packages (3.1.0)
Collecting flask-principal
  Downloading Flask-Principal-0.4.0.tar.gz (5.5 kB)
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: Werkzeug>=3.1 in c:\users\shrut\appdata\roaming\python\python312\site-packages (from flask) (3.1.3)
Requirement already satisfied: Jinja2>=3.1.2 in c:\users\shrut\appdata\roaming\python\python312\site-packages (from flask) (3.1.5)
Requirement already satisfied: itsdangerous>=2.2 in c:\users\shrut\appdata\roaming\python\python312\site-packages (from flask) (2.2.0)
Requirement already satisfied: click>=8.1.3 in c:\users\shrut\appdata\roaming\python\python312\site-packages (from flask) (8.1.8)
Requirement already satisfied: blinker>=1.9 in c:\users\shrut\appdata\roaming\python\python312\site-packages (from flask) (1.9.0)
Requirement already satisfied: colorama in c:\users\shrut\appdata\roaming\python\python312\site-packages (from click>=8.1.3->flask) (0.4.6)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\shrut\appdata\roaming\python\python312\site-packages (from Jinja2>=3.1.2->flask) (3.0.2)
Building wheels for collected packages: flask-principal
  Building wheel for flask-principal (pyproject.toml) ... done
  Created wheel for flask-principal: filename=flask_principal-0.4.0-py3-none-any.whl size=5645 sha256=3e6c62341df4dacd0f3abaffbc8ccb7e1f2a7e77ba793a522636ab
9227217f66
  Stored in directory: c:\users\shrut\appdata\local\pip\cache\wheels\88\f9\fa\de59e4a5a9d8c319c22f841229ee4563907603d8ce819ea52b
Successfully built flask-principal
Installing collected packages: flask-principal
Successfully installed flask-principal-0.4.0

[notice] A new release of pip is available: 24.2 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\shrut>python --version
Python 3.12.7
```

Step 2: Set Up the Project Directory

Create a project folder where we will store our access control implementations.



```
C:\Users\shrut>mkdir access_control

C:\Users\shrut>cd access_control
```

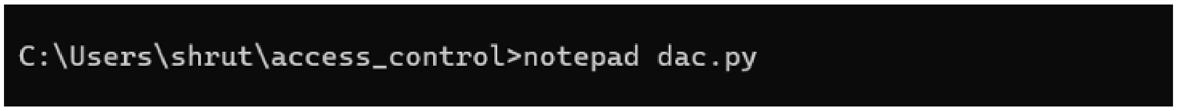
This will:
1. Create a new folder called access_control.
2. Change the working directory to access_control.

Step 3: Implement Discretionary Access Control (DAC)

Discretionary Access Control (DAC) is a type of access control where resource owners determine who can access specific resources. In this case, we will use role-based permissions to define who can access what.

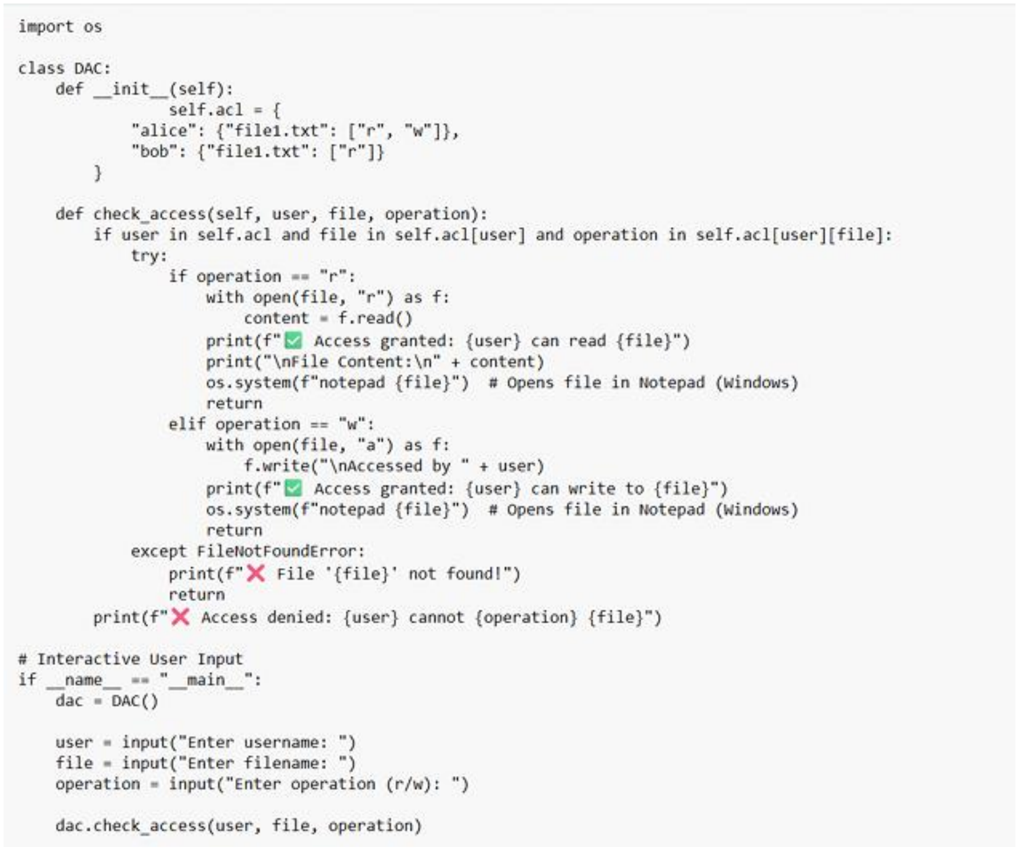
Create the DAC Script

1. Open Notepad (or any text editor) and create a new file called dac.py:



```
C:\Users\shrut\access_control>notepad dac.py
```

2. Write the following Python code in dac.py:



```python
import os

class DAC:
    def __init__(self):
                self.acl = {
            "alice": {"file1.txt": ["r", "w"]},
            "bob": {"file1.txt": ["r"]}
        }

    def check_access(self, user, file, operation):
        if user in self.acl and file in self.acl[user] and operation in self.acl[user][file]:
            try:
                if operation == "r":
                    with open(file, "r") as f:
                        content = f.read()
                    print(f"✅ Access granted: {user} can read {file}")
                    print("\nFile Content:\n" + content)
                    os.system(f"notepad {file}")  # Opens file in Notepad (Windows)
                    return
                elif operation == "w":
                    with open(file, "a") as f:
                        f.write("\nAccessed by " + user)
                    print(f"✅ Access granted: {user} can write to {file}")
                    os.system(f"notepad {file}")  # Opens file in Notepad (Windows)
                    return
            except FileNotFoundError:
                print(f"❌ File '{file}' not found!")
                return
        print(f"❌ Access denied: {user} cannot {operation} {file}")

# Interactive User Input
if __name__ == "__main__":
    dac = DAC()

    user = input("Enter username: ")
    file = input("Enter filename: ")
    operation = input("Enter operation (r/w): ")

    dac.check_access(user, file, operation)
```

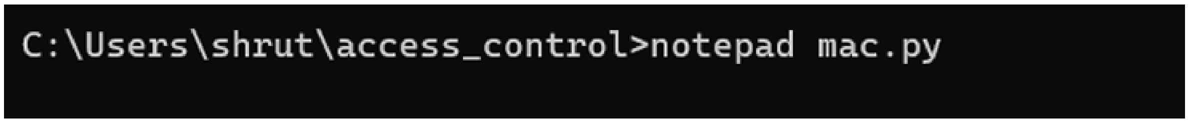
In the code above:
1. Defines a DAC class with an Access Control List (ACL) mapping users to file permissions.
2. Checks if a user has the required permission to read (r) or write (w) a file.
3. If read (r) permission exists, it opens the file, prints its content, and launches Notepad.
4. If write (w) permission exists, it appends a log entry and opens the file in Notepad.
5. Handles file not found errors to prevent crashes.
6. Accepts user input for username, filename, and operation, then validates access accordingly.

Step 4: Implement Mandatory Access Control (MAC)

Mandatory Access Control (MAC) is a security model where access permissions are determined by security policies rather than user discretion. Users have clearance levels, and resources have classification levels.

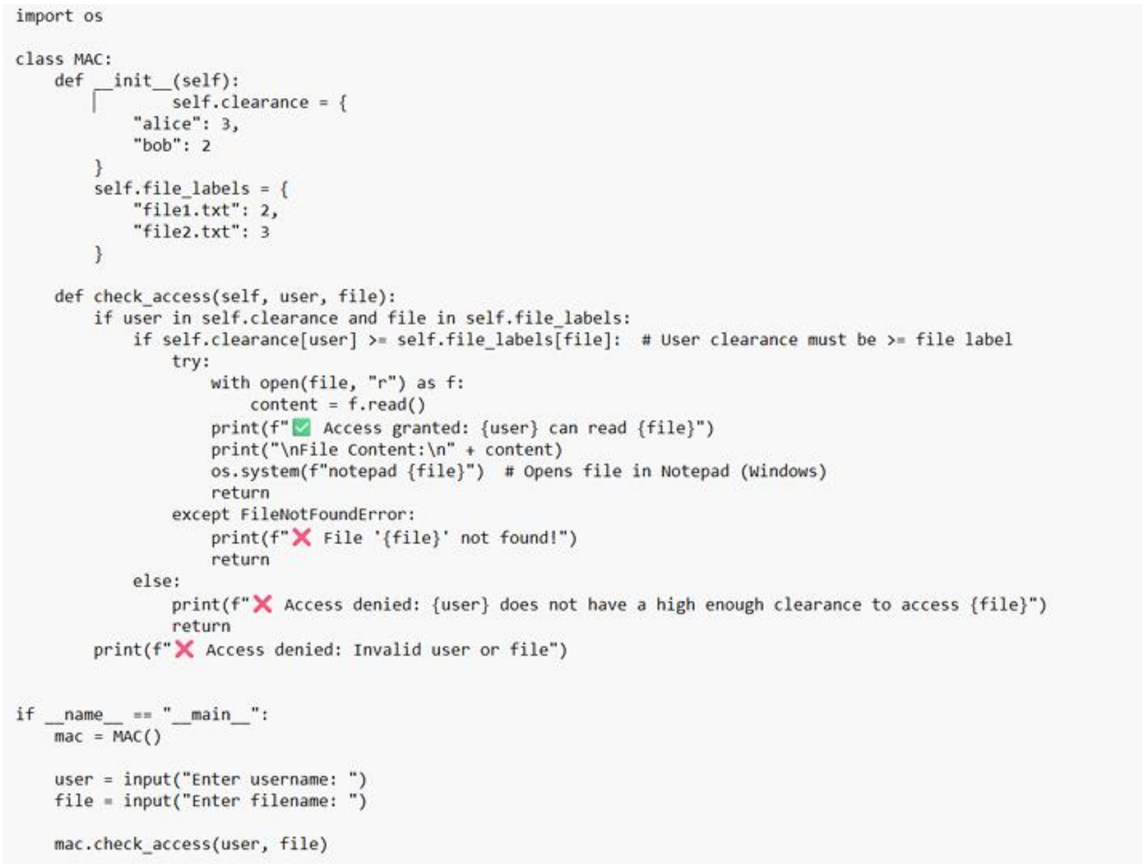
Create the MAC Script

1. Open Notepad and create a new file called mac.py:



```
C:\Users\shrut\access_control>notepad mac.py
```

2. Write the following Python code in mac.py:

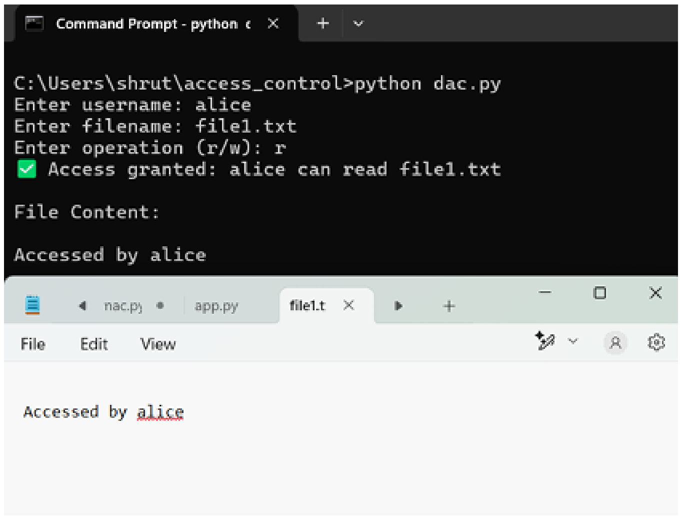


```
import os

class MAC:
    def __init__(self):
        |       self.clearance = {
            "alice": 3,
            "bob": 2
        }
        self.file_labels = {
            "file1.txt": 2,
            "file2.txt": 3
        }

    def check_access(self, user, file):
        if user in self.clearance and file in self.file_labels:
            if self.clearance[user] >= self.file_labels[file]:  # User clearance must be >= file label
                try:
                    with open(file, "r") as f:
                        content = f.read()
                    print(f"✅ Access granted: {user} can read {file}")
                    print("\nFile Content:\n" + content)
                    os.system(f"notepad {file}")  # Opens file in Notepad (Windows)
                    return
                except FileNotFoundError:
                    print(f"❌ File '{file}' not found!")
                    return
            else:
                print(f"❌ Access denied: {user} does not have a high enough clearance to access {file}")
                return
        print(f"❌ Access denied: Invalid user or file")


if __name__ == "__main__":
    mac = MAC()

    user = input("Enter username: ")
    file = input("Enter filename: ")

    mac.check_access(user, file)
```

In the code above:
1. Defines a MAC class that enforces access control based on security clearance levels.
2. Maintains a clearance dictionary where each user has a clearance level.
3. Assigns security labels to files, determining their required clearance level.
4. Checks if the user's clearance is high enough to read a requested file.
5. If access is granted, it opens the file, prints its content, and launches Notepad.
6. Handles file not found errors and denies access for invalid users or insufficient clearance.

Step 5: Run the DAC and MAC script

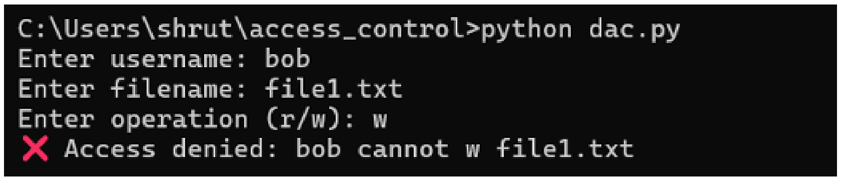
1. After saving dac.py, execute the following command:



```
C:\Users\shrut\access_control>python dac.py
Enter username: alice
Enter filename: file1.txt
Enter operation (r/w): r
✅ Access granted: alice can read file1.txt

File Content:

Accessed by alice
```

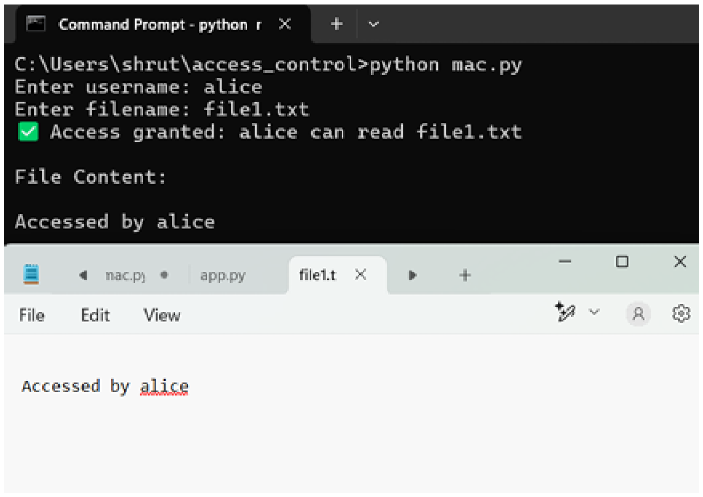
```
Accessed by alice
```

2. If the credentials do not match with the permission, then the output will be:



```
C:\Users\shrut\access_control>python dac.py
Enter username: bob
Enter filename: file1.txt
Enter operation (r/w): w
❌ Access denied: bob cannot w file1.txt
```

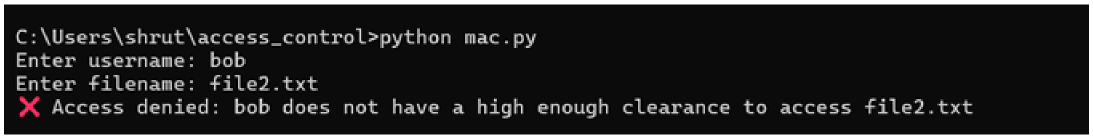
3. After saving the mac.py file, run the following command:



```
C:\Users\shrut\access_control>python mac.py
Enter username: alice
Enter filename: file1.txt
✅ Access granted: alice can read file1.txt

File Content:

Accessed by alice
```

```
Accessed by alice
```

4. If the credentials do not match, then the output is:



```
C:\Users\shrut\access_control>python mac.py
Enter username: bob
Enter filename: file2.txt
❌ Access denied: bob does not have a high enough clearance to access file2.txt
```

Conclusion

The implementation of Discretionary Access Control (DAC) and Mandatory Access Control (MAC) demonstrates two distinct approaches to securing information. DAC provides a flexible system where access rights are assigned based on user identity and defined permissions. This model allows users to have control over their own files, granting or restricting access as needed. While DAC offers ease of management, it also presents security risks, as unauthorized users may gain access if permissions are not carefully controlled. The system relies on an access control list (ACL) that defines specific read and write privileges for each user, ensuring that only authorized actions are permitted. MAC enforces a more rigid security structure by assigning security clearance levels to both users and files. Access is granted only if a user's clearance level meets or exceeds the classification of the file, ensuring strict adherence to security policies. This model is widely used in environments where data confidentiality is a priority, such as government and military systems. Unlike DAC, MAC does not allow users to modify access permissions, reducing the risk of unauthorized data exposure. By implementing both DAC and MAC, the strengths of each model are leveraged to create a comprehensive security framework that balances flexibility and strict access control.

8. Working with Sniffers for monitoring network communication.