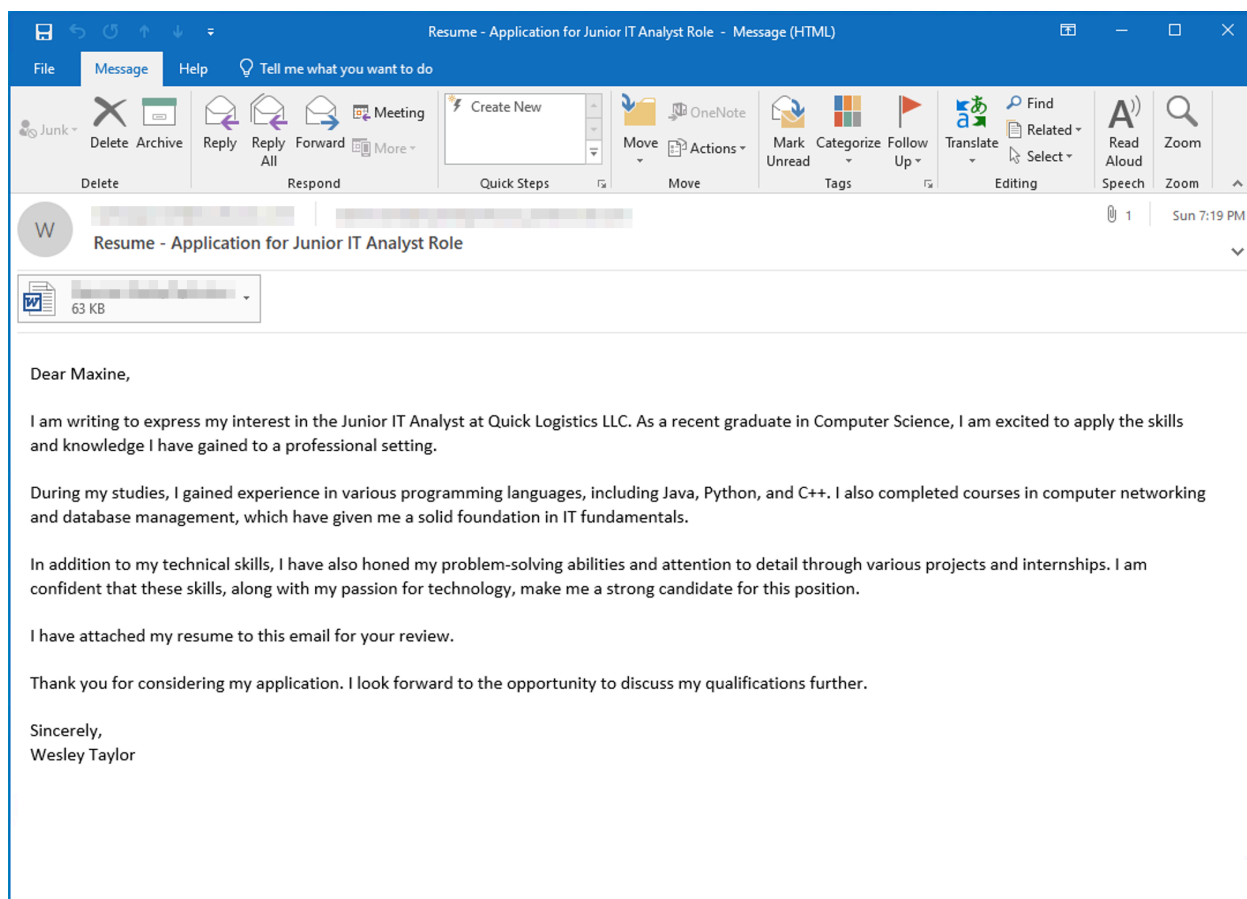


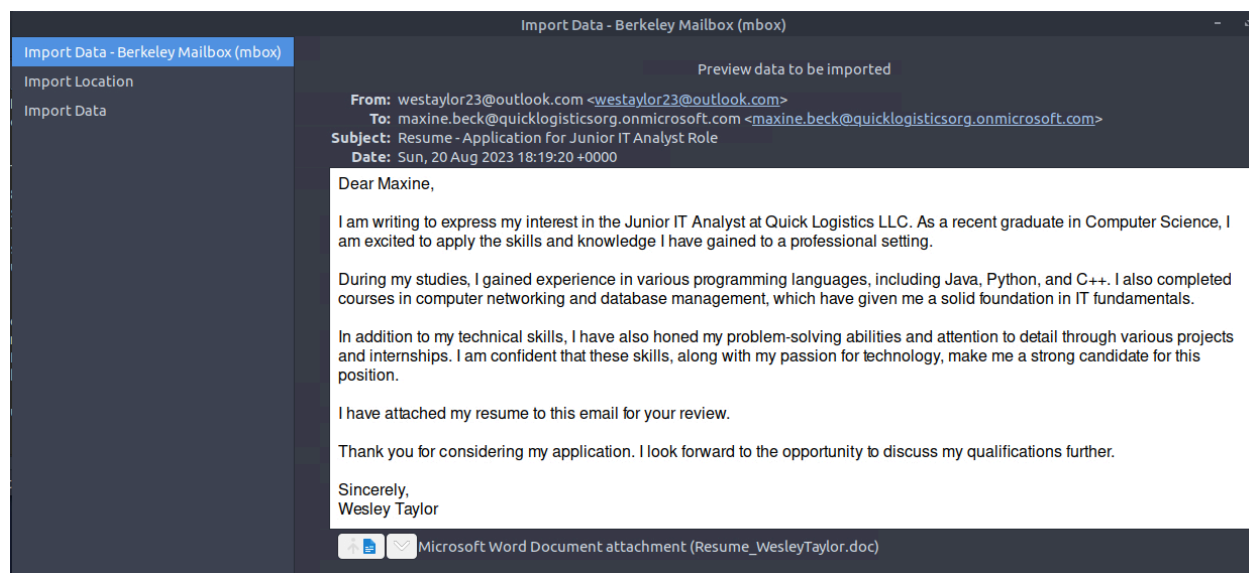
# Intro

The Boogeyman is back!

Maxine, a Human Resource Specialist working for Quick Logistics LLC, received an application from one of the open positions in the company. Unbeknownst to her, the attached resume was malicious and compromised her workstation.



The security team was able to flag some suspicious commands executed on the workstation of Maxine, which prompted the investigation. Given this, you are tasked to analyse and assess the impact of the compromise.



We will be using this screenshot for questions 1-3

Q1- What email was used to send the phishing email?

- I began by importing the provided .mbox file into a mail viewer to examine the headers.
- I identified the external sender address in the "From" field.
- Answer: westaylor23@outlook.com

Q2- What is the email of the victim employee?

- While reviewing the same email headers, I checked the "To" field to identify the targeted department.
- The email was sent to Maxine Beck in Human Resources.
- Answer: maxine.beck@quicklogisticsorg.onmicrosoft.com

Q3-What is the name of the attached malicious document?

- I looked at the bottom of the email to identify any suspicious files Maxine may have interacted with.
- The email contained a Microsoft Word attachment disguised as a resume.
- Answer: Resume\_WesleyTaylor.doc

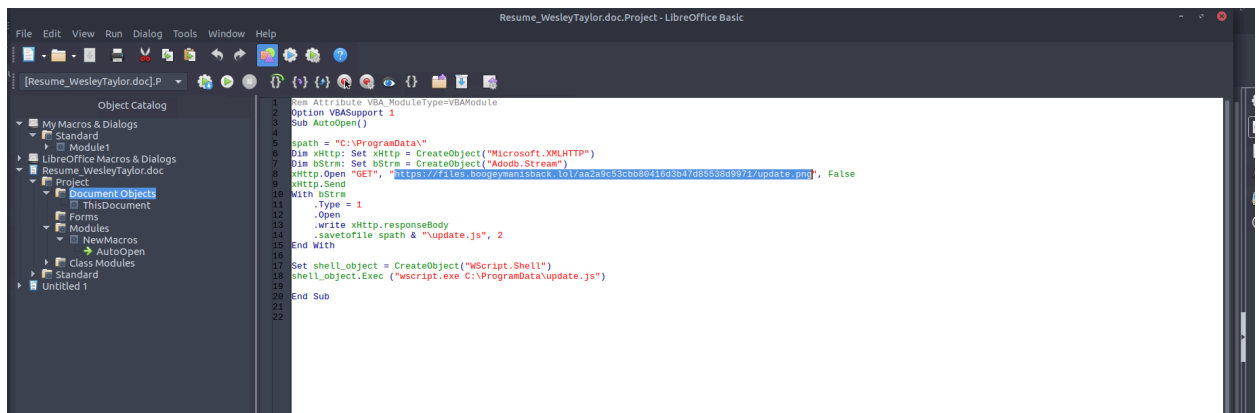
Q4- What is the MD5 hash of the malicious attachment?

- To create a signature for this file, I moved the document to a Linux terminal and utilized the md5sum utility.
- Answer: 52c4384a0b9e248b95804352ebec6c5b

```
ubuntu@tryhackme:~/Desktop/Artefacts$ ls
'Resume - Application for Junior IT Analyst Role.eml' Resume_WesleyTaylor.doc WKSTN-2961.raw
ubuntu@tryhackme:~/Desktop/Artefacts$ md5sum Resume_WesleyTaylor.doc
52c4384a0b9e248b95804352ebec6c5b Resume_WesleyTaylor.doc
```

Q5- What URL is used to download the stage 2 payload based on the document's macro?

- I opened the document in a secure VM using LibreOffice Basic to inspect the AutoOpen() macro.
- I found an xHttp.Open command pointing to an external domain.
- Answer:  
<https://files.boogeymanisback.lol/aa2a9c53cbb80416d3b47d85538d9971/update.png>



Q6- What is the name of the process that executed the newly downloaded stage 2 payload?

- I analyzed the final lines of the macro to see how the "image" file was handled after being saved to the system.
- The macro utilized a built-in Windows script engine to run the file.
- Answer: wscript.exe

Q7- What is the full file path of the malicious stage 2 payload?

- This discovery gave me a specific "Indicator of Compromise" (IOC) to hunt for in the workstation's memory
- I looked at the savetofile argument in the macro to see where the attacker hid the payload.
- Answer: C:\ProgramData\update.js

Q8- What is the PID of the process that executed the stage 2 payload?

- I transitioned to the memory dump (WKSTN-2961.raw) and ran windows.pstree to see which processes were active.
- I searched for wscript.exe and identified its unique process ID.
- Answer: 4260

```
ubuntu@tryhackme:~/Desktop/Artefacts$ vol -f WKSTN-2961.raw windows.pstree.PsTree | grep -i "wscript.exe"
**** 4260 100.01124 wscript.exe 0xe58f864ca0c0 6 - 3 False 2023-08-21 14:12:47.000000 N/A
```

Q9- What is the parent PID of the process that executed the stage 2 payload?

- Using the same pstree output, I looked one level up to see which application originally spawned the malicious script engine.
- The parent was WINWORD.EXE, confirming the attack started when Maxine opened the resume.
- Answer: 1124

Q10- What URL is used to download the malicious binary executed by the stage 2 payload?

- Investigative Step: Knowing the script was running, I looked for a final "Stage 3" download URL.
- Answer:  
<https://files.boogeymanisback.lol/aa2a9c53cbb80416d3b47d85538d9971/updater.exe>

Q11-What is the PID of the malicious process used to establish the C2 connection?

I ran the windows.netscan module to find any processes talking to external IP addresses. I found an active entry for updater.exe.

Answer: 6216

- Ran the command “vol -f WKSTN-2961.raw windows.cmdline.CmdLine” and was able to determine the path and pid

```
5592 SearchProtocol Process 6592: Required memory at 0x1582ef2ce000 is not valid (incomplete layer memory_layer?)
4260 wscript.exe wscript.exe C:\ProgramData\update.js
6216 updater.exe "C:\Windows\Tasks\updater.exe"
```

Q12- What is the full file path of the malicious process used to establish the C2 connection?

Investigative Step: I ran windows.cmdline to see the exact execution path of PID 6216.  
 Answer: C:\Windows\Tasks\updater.exe

Q13-

- Investigative Step: I returned to the netscan results and examined the destination IP and port for the updater.exe process.
- Answer: 128.199.95.189:8080

10.10.49.181	63331	128.199.95.189	8080	CLOSED	6216	updater.exe	2023-08-21 14:15:17.000000
10.10.49.181	63308	128.199.95.189	8080	CLOSED	6216	updater.exe	2023-08-21 14:14:39.000000
10.10.49.181	63291	128.199.95.189	8080	CLOSED	6216	updater.exe	2023-08-21 14:14:13.000000
10.10.49.181	63242	20.189.173.10	443	CLOSED	1124	WINWORD.EXE	2023-08-21 14:12:39.000000
10.10.49.181	63243	20.189.173.10	443	CLOSED	1124	WINWORD.EXE	2023-08-21 14:12:39.000000
10.10.49.181	63348	128.199.95.189	8080	CLOSED	6216	updater.exe	2023-08-21 14:16:05.000000

Q14- What is the full file path of the malicious email attachment based on the memory dump?

- Investigative Step: I searched the command line history of WINWORD.EXE to find the exact location where Outlook saved the attachment.
- Answer:  
 C:\Users\maxine.beck\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\WQHGCZCF\Resume\_WesleyTaylor (002).doc

1440	OUTLOOK.EXE	"C:\Program Files\Microsoft Office\Root\Office16\OUTLOOK.EXE" /ml "C:\Users\maxine.beck\Desktop\Resume - Application for Junior IT Analyst Role.eml"
1124	WINWORD.EXE	"C:\Program Files\Microsoft Office\Root\Office16\WINWORD.EXE" /n "C:\Users\maxine.beck\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\WQHGCZCF\Resume_WesleyTaylor (002).doc"
6720	SearchFilterHost.exe	Process 6720: Required memory at 0x700000000 is not valid (incomplete layer memory_layer?)
4386	WINWORD.EXE	Required memory at 0x6e0370020 is not valid (process exited?)
4776	WinPrvSE.exe	C:\Windows\system32\wbem\WinPrvSE.exe
6592	SearchProtocolHost.exe	Process 6592: Required memory at 0x1592ef2ce000 is not valid (incomplete layer memory_layer?)
4260	wscript.exe	wscript.exe C:\ProgramData\update.js
6216	updater.exe	C:\Windows\Tasks\updater.exe
4464	conhost.exe	?C:\Windows\system32\conhost.exe 0x4
5332	RunDll.exe	RunDll.exe

Q15- The attacker implanted a scheduled task right after establishing the c2 callback. What is the full command used by the attacker to maintain persistent access?

- Investigative Step: I searched the memory strings for schtasks to find how the attacker scheduled the malware to survive a reboot.
- Answer: schtasks /Create /F /SC DAILY /ST 09:00 /TN Updater /TR  
 'C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -NonI -W hidden -c \"IEX ([Text.Encoding]::UNICODE.GetString([Convert]::FromBase64String((gp HKCU:\Software\Microsoft\Windows\CurrentVersion debug).debug)))\"

```

obuntu@tryhackme:~/Desktop/Artefacts$ strings MKSTN-2961.raw | grep -i "schtasks"
;.run "cmd.exe /c echo " & chr(powershell.exe [io.file]::writeallbytes(schtasks) /create /f /sc minute /mo 3 /tn.run "cmd.exe /c echo " & "set
Vad schtasks v
& schtasks /Delete
wb ckAFABJAEUAMAA=schtasks /cre
"cmd /c schtasks /Run /TN
schtasks+
schtasks /Create /tn "xs"0
FFileWithschtasks.A
), "0" schtasks /crt
htasks /create /sc minuq
* htasks /cre
htasks
un"schtasks /cre
schtasks:minute+
schtasks.exe /CREATE /RL H
schtasks /
Nonstdschtasks
schtasks.exe /creat8
schtasks
schtasks
schtasks /CREATE /SC ONLOGON
schtasks.
schtasks.pdb
BKGUiceBZACAAQ08KACQAKAAIAENabwBvAGsAq@IACIALAALAGpAbAGAEsAcwBBAEBAga9APfKAYgBNAEwMwXaAGsAugBtAESaZ0RRADUAMAAZAEBAQABHAGsAcw44FCA0ABXADQAZGRZADBA1gApADsA3ABKACEAdABHADOA3AB3ACHALgBEACBAdwBuAGwBwBhACQARABHA
HQBVOAqACDQACwBIAHTAKWAKAHQAKQATACQaaQB2AD0A2ABKAGEAdABHAFsAMAAUAC4AhwBdAdS2ABKAGEAdABHADOA3ABKAGEAdABHAFsANAAUAC4A3ABKAGEAdABHAC4ABAB1AG4AZwB0AGgAXQATAC0AgBvAGkAbpBbAFEMAAABHAIHwBdAFDAKAAPACAA3AB5ACAA3ABKAGEA
dABHACAAKAAKAEKAVGAFACQASwAPaCKAFABJAEUAMAA=schtasks /Create /F /SC DAILY /ST 09:00 /TN Updater /TR "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -NonI -W hidden -c "[IEX ([Text.Encoding]::Unicode)
GetString((ConvertFromBase64String((cp HKCU\Software\Microsoft\Windows\CurrentVersion\debug).debug))))]"schtasks persistence established using listener http stored in HKCU:\Software\Microsoft\Windows\Curre
ntVersion\debug with Updater daily trigger at 09:00"
PSPParameterBinding(Out-String): name="InputObject"; value="schtasks persistence established using listener http stored in HKCU:\Software\Microsoft\Windows\CurrentVersion\debug with Updater daily trigger at 09:0
0."
schtasks persistence established using listener http stored in HKCU:\Software\Microsoft\Windows\CurrentVersion\debug with Updater daily trigger at 09:00.
schtasks /
GTSschtasks.exe /creat8
schtasks persistence established using listener http stored in HKCU:\Software\Microsoft\Windows\CurrentVersion\debug with Updater daily trigger at 09:00.
schtasks PA
schtasks.w

```

## Recap

Through Log Pivoting, I traced the attack from a single email (westaylor23@outlook.com) to a weaponized document, recovered the hidden JavaScript payload (update.js), and finally identified the C2 server and the persistence mechanism designed to run daily at 09:00.

This lab demonstrates my ability to reconstruct a complex attack chain using volatile memory and static document analysis.