AutoIT Bot Targets Gmail Accounts First

blog.sonicwall.com/en-us/2024/08/autoit-bot-targets-gmail-accounts-first/

Security News August 27, 2024

Summary

This week, the SonicWall Capture Labs threat research team observed an AutoIT-compiled executable that attempts to open Gmail login pages via MS Edge, Google Chrome and Mozilla Firefox. It has functionality to read clipboard data, capture keystrokes, run as different users, and restart or shutdown the system. The sample is also capable of detecting debuggers and blocking user input if one is detected, as well as directing control of keyboard and mouse events. It is imperative to be cautious when running files of unknown origin or with vague names such as “File.exe”. SonicWall customers are protected in the daily update feed via the “MalAgent.AutoITBot” signature.

Technical Analysis

Using the Detect-It-Easy (DIE) tool to review a sample shows the malware as an AutoIT executable. Note the original name was “File.exe”.

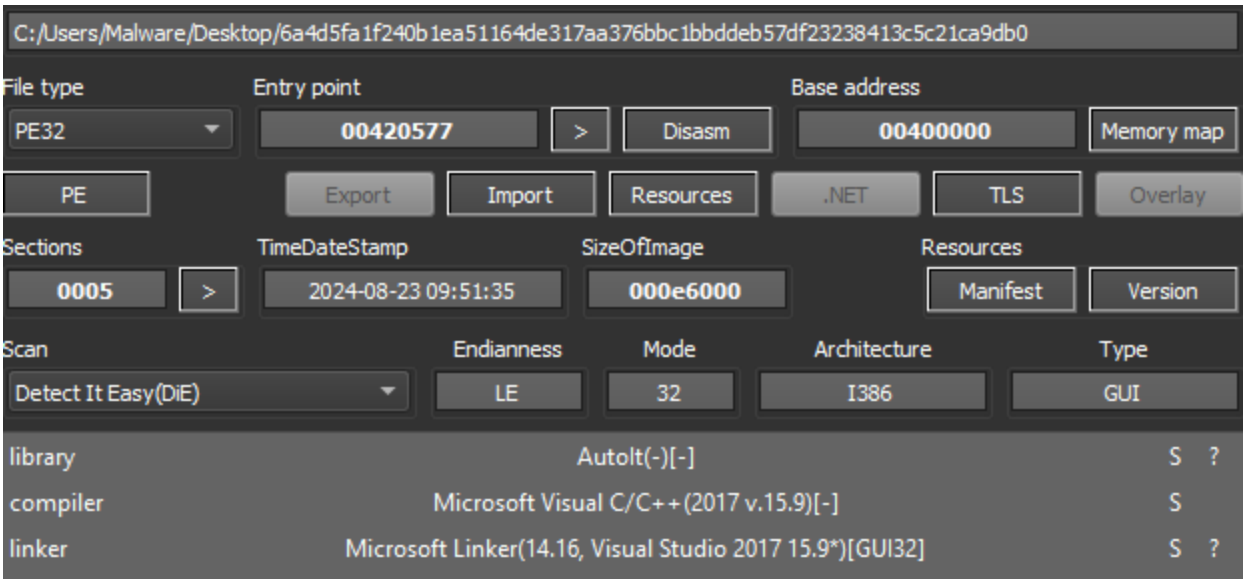
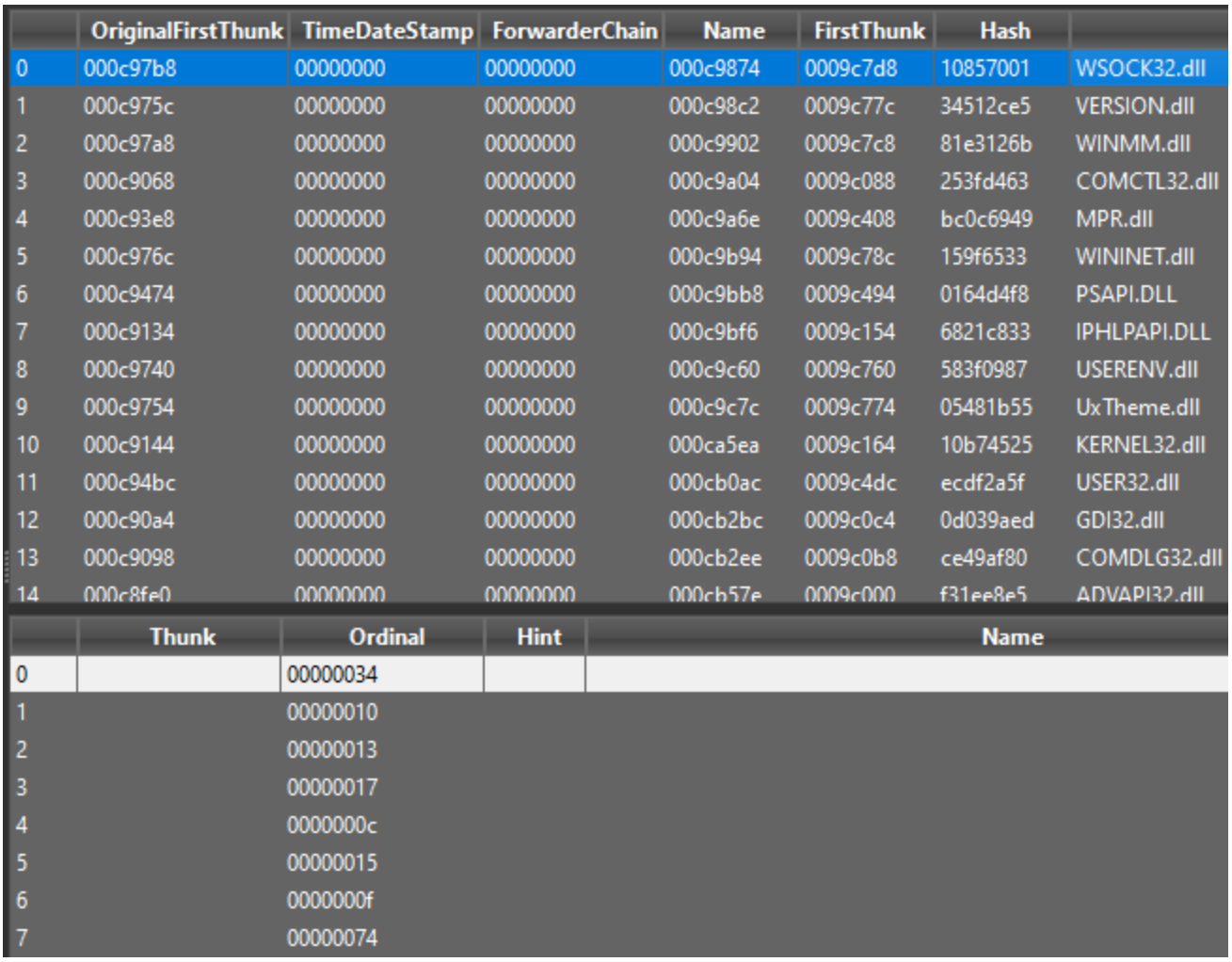


Figure 1: DIE Sample detection

Multiple libraries are being imported with no data outside of ordinals identifying the related functions, as well as four separate networking libraries. This indicates the libraries have been obfuscated, and it can be seen by using the DIE tool in Figure 2.



	OriginalFirstThunk	TimeDateStamp	ForwarderChain	Name	FirstThunk	Hash	
0	000c97b8	00000000	00000000	000c9874	0009c7d8	10857001	WSOCK32.dll
1	000c975c	00000000	00000000	000c98c2	0009c77c	34512ce5	VERSION.dll
2	000c97a8	00000000	00000000	000c9902	0009c7c8	81e3126b	WINMM.dll
3	000c9068	00000000	00000000	000c9a04	0009c088	253fd463	COMCTL32.dll
4	000c93e8	00000000	00000000	000c9a6e	0009c408	bc0c6949	MPR.dll
5	000c976c	00000000	00000000	000c9b94	0009c78c	159f6533	WININET.dll
6	000c9474	00000000	00000000	000c9bb8	0009c494	0164d4f8	PSAPI.DLL
7	000c9134	00000000	00000000	000c9bf6	0009c154	6821c833	IPHLPAPI.DLL
8	000c9740	00000000	00000000	000c9c60	0009c760	583f0987	USERENV.dll
9	000c9754	00000000	00000000	000c9c7c	0009c774	05481b55	UxTheme.dll
10	000c9144	00000000	00000000	000ca5ea	0009c164	10b74525	KERNEL32.dll
11	000c94bc	00000000	00000000	000cb0ac	0009c4dc	ecdf2a5f	USER32.dll
12	000c90a4	00000000	00000000	000cb2bc	0009c0c4	0d039aed	GDI32.dll
13	000c9098	00000000	00000000	000cb2ee	0009c0b8	ce49af80	COMDLG32.dll
14	000c8fe0	00000000	00000000	000cb57e	0009c000	f31ee8e5	ADVAPI32.dll

	Thunk	Ordinal	Hint	Name
0		00000034		
1		00000010		
2		00000013		
3		00000017		
4		0000000c		
5		00000015		
6		0000000f		
7		00000074		

Figure 2: Obfuscated libraries

Using the AutoITExtractor tool we can extract the script shown in Figure 3. This allows us to see it has cleartext commands to find and launch each browser on a Google sign in page (accounts.google.com)

```
Local $url = "https://accounts.google.com/ServiceLogin?service=accountsettings&continue=https://accounts.goo
Local $chromepath = ""
Local $edgepath = ""
Local $firefoxpath = ""
Local $programfiles64 = "C:\Program Files"
Local $programfiles86 = "C:\Program Files (x86)"
Local $userspath = @HomeDrive & "\Users"
Func _CHECKINSTANDARDDIRS($apppath, ByRef $browserpath)
    If FileExists($apppath) Then
        $browserpath = $apppath
        Return 0x1
    EndIf
    Return 0x0
EndFunc   ;==>_CHECKINSTANDARDDIRS
Func _CHECKINUSERPROFILES($relativepath, ByRef $browserpath)
    Local $search = FileFindFirstFile($userspath & "\*")
    If $search = +0xffffffff Then Return
    While 0x1
        Local $userfolder = FileFindNextFile($search)
        If @error Then ExitLoop
        If StringInStr($userfolder, ".") = 0x0 Then
            Local $fullpath = $userspath & "\" & $userfolder & "\AppData\Local\" & $relativepath
            If FileExists($fullpath) Then
                $browserpath = $fullpath
                ExitLoop
            EndIf
        EndIf
    WEnd
    FileClose($search)
EndFunc   ;==>_CHECKINUSERPROFILES
If Not _CHECKINSTANDARDDIRS($programfiles64 & "\Google\Chrome\Application\chrome.exe", $chromepath) Then
    If Not _CHECKINSTANDARDDIRS($programfiles86 & "\Google\Chrome\Application\chrome.exe", $chromepath) Then
        _CHECKINUSERPROFILES("Google\Chrome\Application\chrome.exe", $chromepath)
    EndIf
EndIf
If Not _CHECKINSTANDARDDIRS($programfiles64 & "\Microsoft\Edge\Application\msedge.exe", $edgepath) Then
    If Not _CHECKINSTANDARDDIRS($programfiles86 & "\Microsoft\Edge\Application\msedge.exe", $edgepath) Then
        _CHECKINUSERPROFILES("Microsoft\Edge\Application\msedge.exe", $edgepath)
    EndIf
EndIf
If Not _CHECKINSTANDARDDIRS($programfiles64 & "\Mozilla Firefox\firefox.exe", $firefoxpath) Then
    If Not _CHECKINSTANDARDDIRS($programfiles86 & "\Mozilla Firefox\firefox.exe", $firefoxpath) Then
        _CHECKINUSERPROFILES("Mozilla Firefox\firefox.exe", $firefoxpath)
    EndIf
EndIf
If $chromepath <> "" And ProcessExists("chrome.exe") Then
```

Figure 3: Extracted script contents

Statically analyzing the binary using a disassembler yields there are no hardcoded addresses that are known to be malicious. While the script has each browser attempt to access Google accounts, there are generic login links for Facebook, Reddit, and other major social media sites. While the browsers launch and execute, a separate function will set up a listening socket if the environment is correct and connectivity has been established as shown in Figure 4.

```
test al,1
je 6a4d5fa1f240b1ea51164de317aa376bbc1bbddeb57df23238413c5c21ca9db
push 4
lea eax,dword ptr ss:[ebp+8]
mov dword ptr ss:[ebp+8],1
push eax
push 20
push FFFF
push dword ptr ss:[ebp+C]
call dword ptr ds:[<&setsockopt>]
test eax,eax
je 6a4d5fa1f240b1ea51164de317aa37
call dword ptr ds:[<&WSAGetLastEr
```

```
6FF913D0 <wsock32.setsockopt>
mov edi,edi
```

Figure 4: Socket option setup

The malware will call the standard WSAGetLastError Windows API, as seen during dynamic analysis, if the socket setup fails, as seen in Figure 5.

call dword ptr [0055C814h]	bind@WSOCK32.DLL (Import, Unknown Params)
cmp eax, FFFFFFFFh	Return Compare (bind)
jne 005412FFh	target: 005412FF
call dword ptr [0055C824h]	WSAGetLastError@WSOCK32.DLL (Import, Unknown Params)

Figure 5: Socket bind operation (failed)

When the browsers are run, they create multiple processes using the following command line structure:

```
-   C:\Program Files\Mozilla Firefox\firefox.exe" -contentproc --channel=2240 -parentBuildID
20230927232528 -prefsHandle 2188 -prefMapHandle 2180 -prefsLen 25308 -prefMapSize 237879 -
win32kLockedDown -appDir "

-   C:\Program Files\Mozilla Firefox\browser" - {b8205cf2-3fc0-4029-8ece-1522148321f6} 5772
"\\.\pipe\gecko-crash-server-pipe.5772" 15d8296dd10 socket

-   \Program Files (x86)\Microsoft\Edge\Application\msedge.exe" --type=utility --utility-sub-
type=asset_store.mojom.AssetStoreService --lang=en-GB --service-sandbox-type=asset_store_service --
mojo-platform-channel-handle=6680 --field-trial-
handle=2664,i,10002122715651088037,8823588314030443706,262144 /prefetch:8
```

Figure 6: Browser command line commands

The first process creates a hidden, separate page in Firefox, while the second attempts to open the socket.

Once a connection is made, the functions for keylogging, screen capture and further file enumeration take place. This behavior was not observed during testing, however, and no connection was made by a C2 server.

SonicWall Protections

To ensure SonicWall customers are protected against this threat, the following signature has been released:

MalAgent.AutoITBot

IOCs

File.exe

6a4d5fa1f240b1ea51164de317aa376bbc1bbddeb57df23238413c5c21ca9db0

Security News

The SonicWall Capture Labs Threat Research Team gathers, analyzes and vets cross-vector threat information from the SonicWall Capture Threat network, consisting of global devices and resources, including more than 1 million security sensors in nearly 200 countries and territories. The research team identifies, analyzes, and mitigates critical vulnerabilities and malware daily through in-depth research, which drives protection for all SonicWall customers. In addition to safeguarding networks globally, the research team supports the larger threat intelligence community by releasing weekly deep technical analyses of the most critical threats to small businesses, providing critical knowledge that defenders need to protect their networks.