# Special Series Analysis on Popular Malicious Loader Families - Part One | XLoader

**Antiy CERT**

*The original report is in Chinese, and this version is an AI-translated edition.*

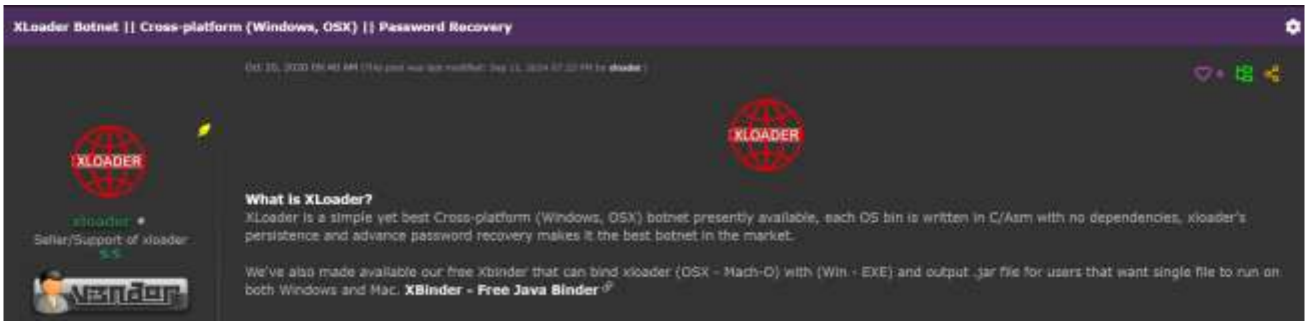
## 1 Introduction

As cyberattack techniques continue to evolve, malicious code loaders have gradually become a key component of malicious code execution. Such loaders are malicious tools used to load various malicious codes into infected systems. They are usually responsible for bypassing system security protections, injecting malicious codes into memory and executing them, laying the foundation for the subsequent deployment of Trojan-type malicious codes. The core functions of loaders include persistence mechanisms, fileless memory execution, and multi-level evasion techniques.

Antiy CERT will compile the information about typical malicious loader families that have been tracked and stored in recent years into a special report, which will be released in sequence in the next few months, and will continue to track new popular loader families. This project will focus on the technical details of the loader and deeply explore its core functions in the attack chain, including its obfuscation technology, encryption mechanism, and injection strategy. In addition, we will continue to improve our own security product capabilities, adopt effective technical solutions to further improve the recognition rate and accuracy of loaders, and help user organizations discover and prevent potential threats in advance.

## 2 Xloader Family Introduction

XLoader is a botnet loader developed from Formbook [1]. It first appeared on hacker forums in 2020 and was sold as Malware-as-a-Service (MaaS). The loader is cross-platform and can run on Windows and macOS operating systems. XLoader **can steal credential information saved in multiple browsers, email clients, and Windows Credential Manager, and can also take desktop screenshots, record clipboard contents, and keyboard keystrokes, posing a serious threat to user privacy.** In addition, XLoader can also execute instructions from attackers to further download and execute other malicious programs, bringing additional risks to user system security.


XLoader Botnet || Cross-platform (Windows, OSX) || Password Recovery

xloader
Seller/Support of xloader

What is XLoader?
XLoader is a simple yet best Cross-platform (Windows, OSX) botnet presently available, each OS bin is written in C/Asm with no dependencies, xloader's persistence and advance password recovery makes it the best botnet in the market.

We've also made available our free Xbinder that can bind xloader (OSX - Mach-O) with (Win - EXE) and output .jar file for users that want single file to run on both Windows and Mac. **XBinder - Free Java Binder**

Figure 2-1 2

**In order to evade detection, XLoader uses a highly complex loading method.** The loader encrypts all of its key codes and data, decrypts them on demand during runtime, and re-encrypts them after use to enhance its concealment. At the same time, XLoader will move itself to other processes through process hollowing to hide its own process. In addition, XLoader will also conduct a detailed inspection of the operating environment for anti-debugging and anti-virtualization, and take anti-hooking measures to prevent key API calls from being hooked, reducing the risk of being automatically analyzed. In order to hide C2, XLoader mixes the real C2 address with a large number of fake addresses, making it difficult to distinguish and further improving its concealment. **It has been verified that Antiy Intelligent Endpoint Protection System (IEP) can effectively detect and kill this botnet loader .**

For more information about the loader, see Antiy Virus Encyclopedia[2].

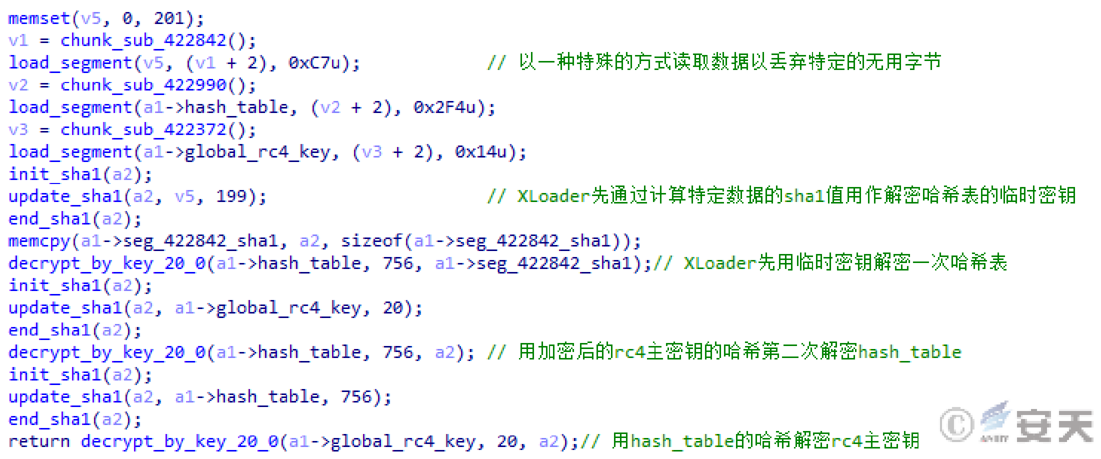
Long press to identify the QR code to view XLoader details

# 3 XLoader Survival Technology Example Analysis

In order to achieve malicious functions, XLoader will improve its survivability from multiple directions. XLoader uses encryption technology to encrypt its own important data and core code to prevent static analysis and memory dump. And through a large number of environmental detection codes, it prevents itself from running in the analysis environment to achieve the purpose of anti-debugging. XLoader also uses anti-hooking technology to further prevent its own API calls from being hooked, further reducing the risk of being monitored. Finally, through injection technology, it can not only hide its own process, but also enter the target program address space to execute malicious logic and carry out malicious behavior. And by modifying the registry to achieve persistence, the purpose of survival is achieved.

## 3.1 Cryptographic Technical Analysis

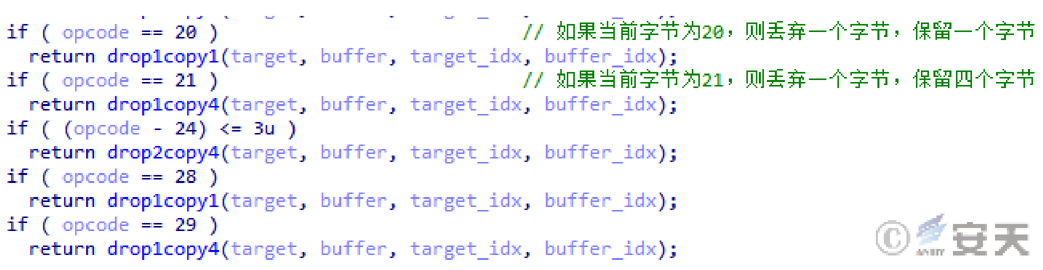
The data in XLoader is encrypted with a key RC 4 key. XLoader uses the sha1 of the hash table used to import functions as a part of the encryption of the RC 4 key to ensure that the hash table has not been tampered with.



```
memset(v5, 0, 201);
v1 = chunk_sub_422842();
load_segment(v5, (v1 + 2), 0xC7u);             // 以一种特殊的方式读取数据以丢弃特定的无用字节
v2 = chunk_sub_422990();
load_segment(a1->hash_table, (v2 + 2), 0x2F4u);
v3 = chunk_sub_422372();
load_segment(a1->global_rc4_key, (v3 + 2), 0x14u);
init_sha1(a2);
update_sha1(a2, v5, 199);                      // XLoader先通过计算特定数据的sha1值用作解密哈希表的临时密钥
end_sha1(a2);
memcpy(a1->seg_422842_sha1, a2, sizeof(a1->seg_422842_sha1));
decrypt_by_key_20_0(a1->hash_table, 756, a1->seg_422842_sha1);// XLoader先用临时密钥解密一次哈希表
init_sha1(a2);
update_sha1(a2, a1->global_rc4_key, 20);
end_sha1(a2);
decrypt_by_key_20_0(a1->hash_table, 756, a2); // 用加密后的rc4主密钥的哈希第二次解密hash_table
init_sha1(a2);
update_sha1(a2, a1->hash_table, 756);
end_sha1(a2);
return decrypt_by_key_20_0(a1->global_rc4_key, 20, a2);// 用hash_table的哈希解密rc4主密钥
```

Figure 3-1 2key

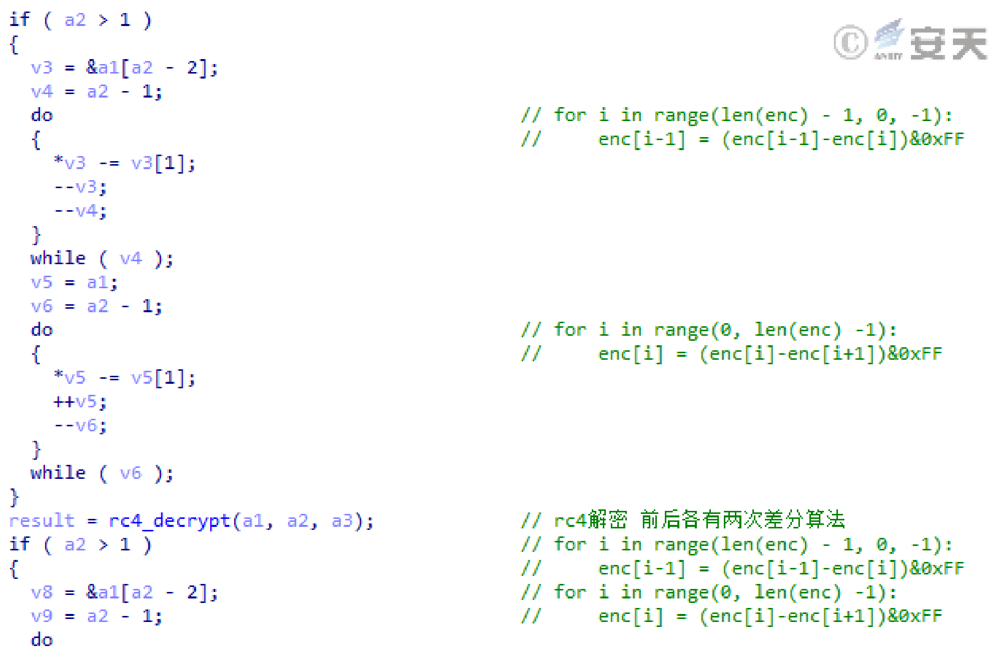
In order to enhance the degree of encryption, XLoader uses a special method to save data. XLoader will mix useless bytes between data and choose the number of bytes to discard and keep according to the byte value read when reading.



```
if ( opcode == 20 )                                    // 如果当前字节为20，则丢弃一个字节，保留一个字节
  return drop1copy1(target, buffer, target_idx, buffer_idx);
if ( opcode == 21 )                                    // 如果当前字节为21，则丢弃一个字节，保留四个字节
  return drop1copy4(target, buffer, target_idx, buffer_idx);
if ( (opcode - 24) <= 3u )
  return drop2copy4(target, buffer, target_idx, buffer_idx);
if ( opcode == 28 )
  return drop1copy1(target, buffer, target_idx, buffer_idx);
if ( opcode == 29 )
  return drop1copy4(target, buffer, target_idx, buffer_idx);
```

Figure 3-3XLoader reads data

During decryption, XLoader will perform decryption operations through differential, RC 4 decryption and then differential to increase the complexity of the encryption function.

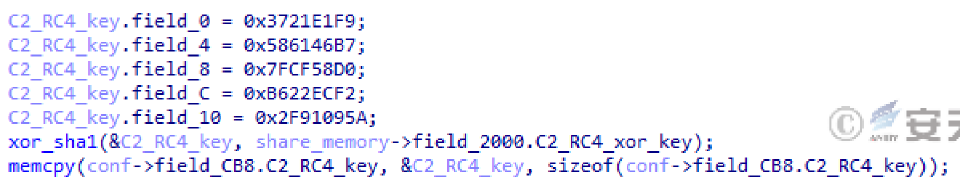


```
if ( a2 > 1 )
{
  v3 = &a1[a2 - 2];
  v4 = a2 - 1;
  do                                    // for i in range(len(enc) - 1, 0, -1):
  {                                     //     enc[i-1] = (enc[i-1]-enc[i])&0xFF
    *v3 -= v3[1];
    --v3;
    --v4;
  }
  while ( v4 );
  v5 = a1;
  v6 = a2 - 1;
  do                                    // for i in range(0, len(enc) -1):
  {                                     //     enc[i] = (enc[i]-enc[i+1])&0xFF
    *v5 -= v5[1];
    ++v5;
    --v6;
  }
  while ( v6 );
}
result = rc4_decrypt(a1, a2, a3);       // rc4解密 前后各有两次差分算法
if ( a2 > 1 )                           // for i in range(len(enc) - 1, 0, -1):
{                                       //     enc[i-1] = (enc[i-1]-enc[i])&0xFF
  v8 = &a1[a2 - 2];                     // for i in range(0, len(enc) -1):
  v9 = a2 - 1;                          //     enc[i] = (enc[i]-enc[i+1])&0xFF
  do
```

Figure 3-4XLoader encryption algorithm

XLoader has additional encryption for the C2 address. In the decrypted string table, the C2 address is represented as base64 encoded. XLoader will decode the base64 at runtime and decrypt it again using the differential and RC4 algorithms.

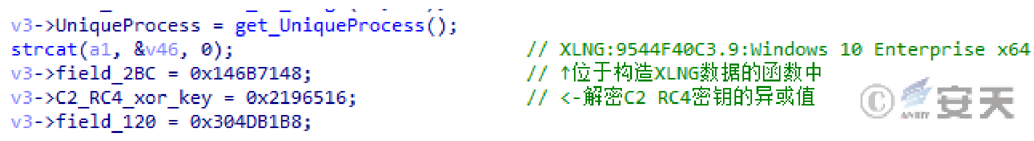
For the RC4 key used to encrypt C2, XLoader encrypts it using an XOR method.



```
C2_RC4_key.field_0 = 0x3721E1F9;
C2_RC4_key.field_4 = 0x586146B7;
C2_RC4_key.field_8 = 0x7FCF58D0;
C2_RC4_key.field_C = 0xB622ECF2;
C2_RC4_key.field_10 = 0x2F91095A;
xor_sha1(&C2_RC4_key, share_memory->field_2000.C2_RC4_xor_key);
memcpy(conf->field_CB8.C2_RC4_key, &C2_RC4_key, sizeof(conf->field_CB8.C2_RC4_key));
```

Figure 3-5XLoader decrypts the C2 RC4 key

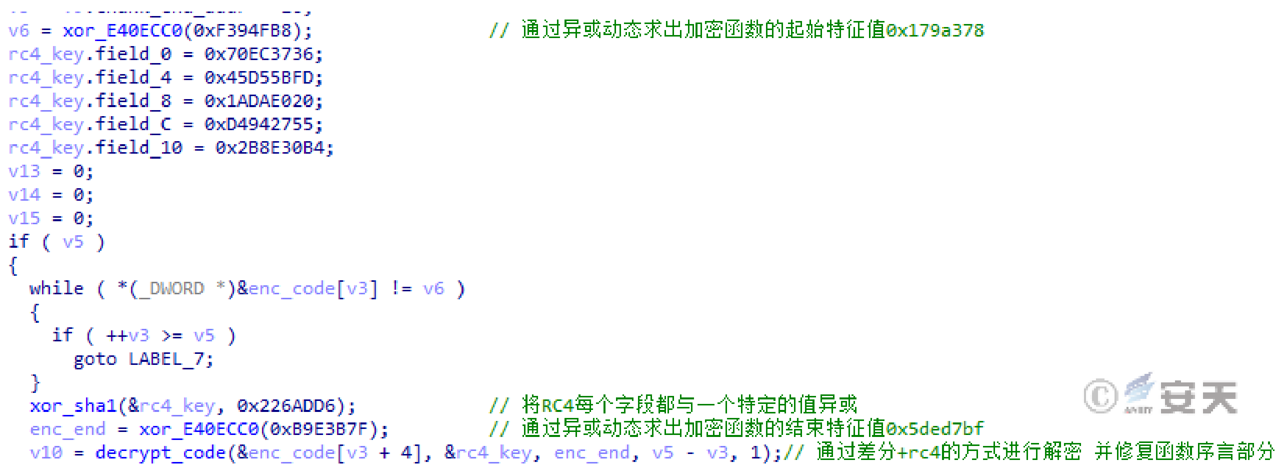
To increase the difficulty of analysis, XLoader encrypts the C2's RC4 key and the XOR key in different functions. The key used for XOR can be found in the function that constructs XLoader XLNG data.



```
v3->UniqueProcess = get_UniqueProcess();
strcat(a1, &v46, 0);                     // XLNG:9544F40C3.9:Windows 10 Enterprise x64:
v3->field_2BC = 0x146B7148;              // ↑位于构造XLNG数据的函数中
v3->C2_RC4_xor_key = 0x2196516;          // <-解密C2 RC4密钥的异或值
v3->field_120 = 0x304DB1B8;
```

Figure 3-6XLoader initializes the XOR key for decrypting C2 RC4

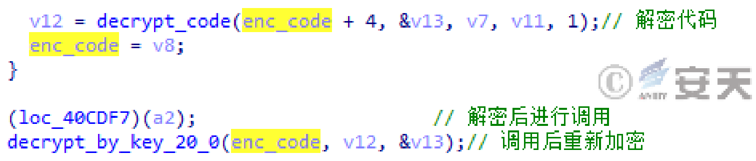
For the code, XLoader uses double RC 4 key encryption. To increase the difficulty of reverse engineering, XLoader will scatter the two decryptions in different locations of the program. To increase the difficulty of extracting the key and encrypted code block, XLoader dynamically calculates the start and end feature values and keys of the encrypted code block at runtime to prevent automated extraction.



```
v6 = xor_E40ECC0(0xF394FB8);             // 通过异或动态求出加密函数的起始特征值0x179a378
rc4_key.field_0 = 0x70EC3736;
rc4_key.field_4 = 0x45D55BFD;
rc4_key.field_8 = 0x1ADAE020;
rc4_key.field_C = 0xD4942755;
rc4_key.field_10 = 0x2B8E30B4;
v13 = 0;
v14 = 0;
v15 = 0;
if ( v5 )
{
  while ( *(_DWORD *)&enc_code[v3] != v6 )
  {
    if ( ++v3 >= v5 )
      goto LABEL_7;
  }
  xor_sha1(&rc4_key, 0x226ADD6);         // 将RC4每个字段都与一个特定的值异或
  enc_end = xor_E40ECC0(0xB9E3B7F);      // 通过异或动态求出加密函数的结束特征值0x5ded7bf
  v10 = decrypt_code(&enc_code[v3 + 4], &rc4_key, enc_end, v5 - v3, 1);// 通过差分+rc4的方式进行解密 并修复函数序言部分
```

Figure 3-7XLoader decrypted code block

To prevent memory dump, XLoader will re-encrypt the code after the function is used.



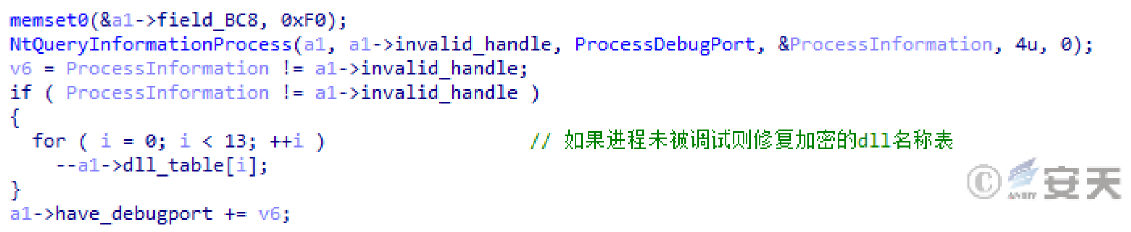
```
  v12 = decrypt_code(enc_code + 4, &v13, v7, v11, 1);// 解密代码
  enc_code = v8;
}

(loc_40CDF7)(a2);                        // 解密后进行调用
decrypt_by_key_20_0(enc_code, v12, &v13);// 调用后重新加密
```

Figure 3-8XLoader function is re-encrypted after use

## 3.2 Anti-Debugging Technology Analysis

XLoader will detect the running environment from multiple aspects to determine whether to continue execution, including: DLL name, DLL path, debug port, kernel debug information, whether WOW32Reserved is hooked, the name of the running process, its own image name and user name.

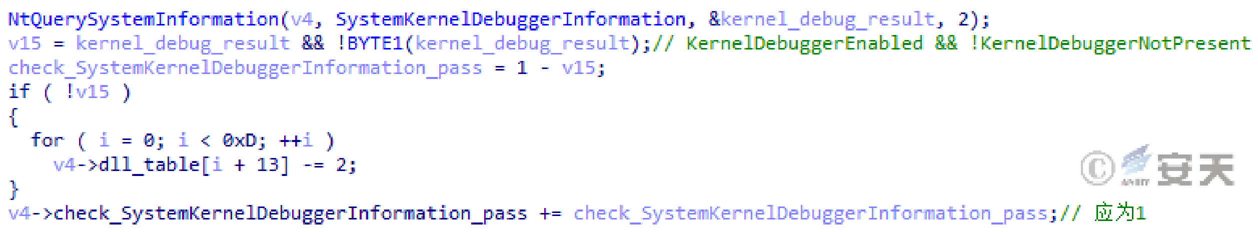
XLoader loader will detect whether it is being debugged through NtQueryInformationProcess .



```
memset0(&a1->field_BC8, 0xF0);
NtQueryInformationProcess(a1, a1->invalid_handle, ProcessDebugPort, &ProcessInformation, 4u, 0);
v6 = ProcessInformation != a1->invalid_handle;
if ( ProcessInformation != a1->invalid_handle )
{
  for ( i = 0; i < 13; ++i )                  // 如果进程未被调试则修复加密的dll名称表
    --a1->dll_table[i];
}
a1->have_debugport += v6;
```

Figure 3-9XLoader detects whether the process is debugged

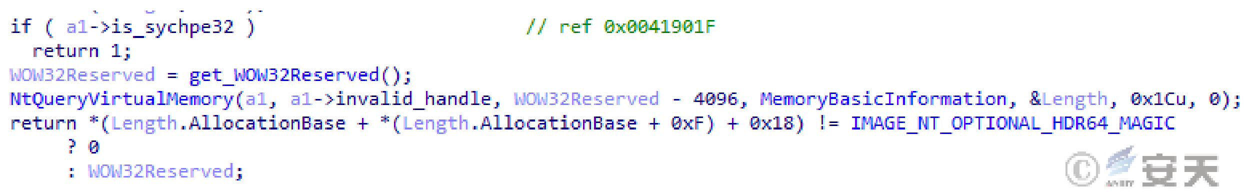
XLoader discovers the kernel debugger through NtQuerySystemInformation .



```
NtQuerySystemInformation(v4, SystemKernelDebuggerInformation, &kernel_debug_result, 2);
v15 = kernel_debug_result && !BYTE1(kernel_debug_result);// KernelDebuggerEnabled && !KernelDebuggerNotPresent
check_SystemKernelDebuggerInformation_pass = 1 - v15;
if ( !v15 )
{
  for ( i = 0; i < 0xD; ++i )
    v4->dll_table[i + 13] -= 2;
}
v4->check_SystemKernelDebuggerInformation_pass += check_SystemKernelDebuggerInformation_pass;// 应为1
```

Figure 3-10XLoader detects whether there is a kernel debugger

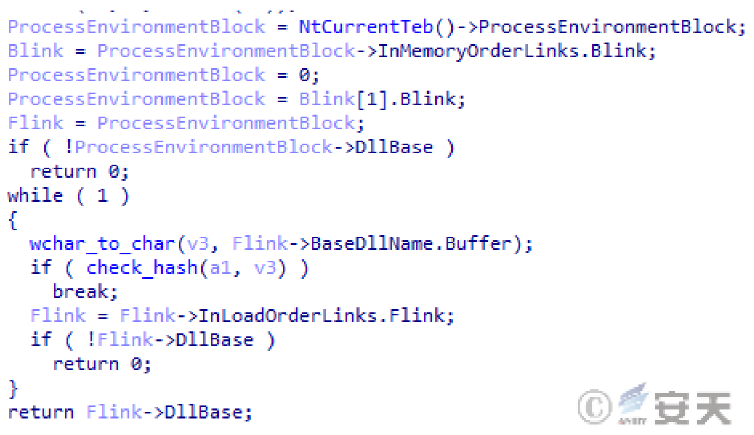
XLoader will detect whether Wow64cpu.dll where WOW 32Reserved is located is 64- bit. If it is 64- bit, it means that there may be API hooks in the current environment and XLoader will stop running.



```
if ( a1->is_sychpe32 )                    // ref 0x0041901F
  return 1;
WOW32Reserved = get_WOW32Reserved();
NtQueryVirtualMemory(a1, a1->invalid_handle, WOW32Reserved - 4096, MemoryBasicInformation, &Length, 0x1Cu, 0);
return *(Length.AllocationBase + *(Length.AllocationBase + 0xF) + 0x18) != IMAGE_NT_OPTIONAL_HDR64_MAGIC
     ? 0
     : WOW32Reserved;
```

Figure 3-11XLoader checks whether Wow64cpu.dll has been modified

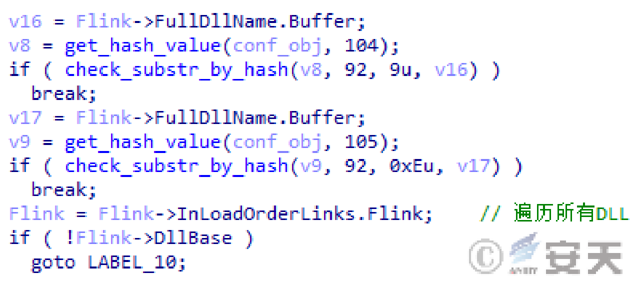
XLoader will look for sbiedll.dll to determine if it is running in the Sandboxie sandbox.



```
ProcessEnvironmentBlock = NtCurrentTeb()->ProcessEnvironmentBlock;
Blink = ProcessEnvironmentBlock->InMemoryOrderLinks.Blink;
ProcessEnvironmentBlock = 0;
ProcessEnvironmentBlock = Blink[1].Blink;
Flink = ProcessEnvironmentBlock;
if ( !ProcessEnvironmentBlock->DllBase )
  return 0;
while ( 1 )
{
  wchar_to_char(v3, Flink->BaseDllName.Buffer);
  if ( check_hash(a1, v3) )
    break;
  Flink = Flink->InLoadOrderLinks.Flink;
  if ( !Flink->DllBase )
    return 0;
}
return Flink->DllBase;
```

Figure 3-12XLoader searches for DLL by traversing the PEB

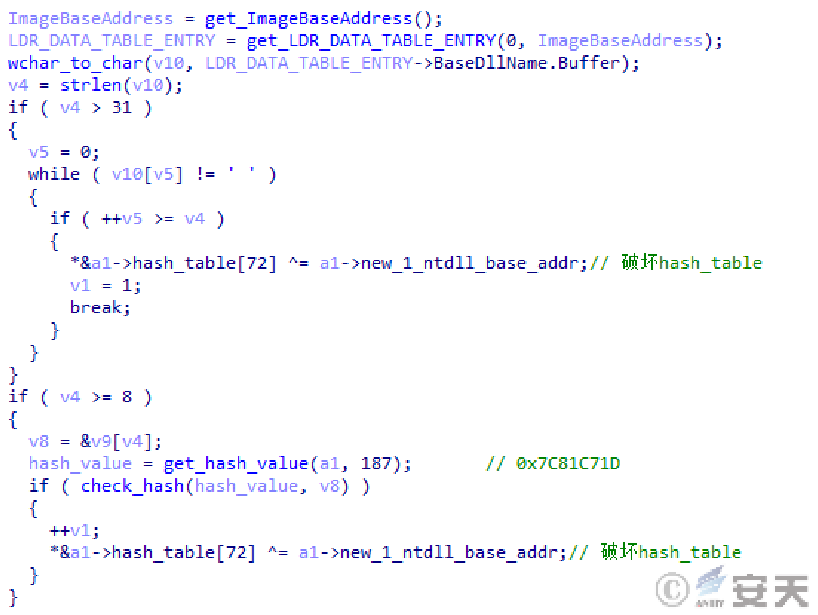
XLoader checks the DLL path to determine if it is running in a sandbox such as Cuckoo and SandCastle .



```
v16 = Flink->FullDllName.Buffer;
v8 = get_hash_value(conf_obj, 104);
if ( check_substr_by_hash(v8, 92, 9u, v16) )
  break;
v17 = Flink->FullDllName.Buffer;
v9 = get_hash_value(conf_obj, 105);
if ( check_substr_by_hash(v9, 92, 0xEu, v17) )
  break;
Flink = Flink->InLoadOrderLinks.Flink;    // 遍历所有DLL
if ( !Flink->DllBase )
  goto LABEL_10;
```

Figure 3-13XLoader checks the DLL to determine whether it is running in a sandbox

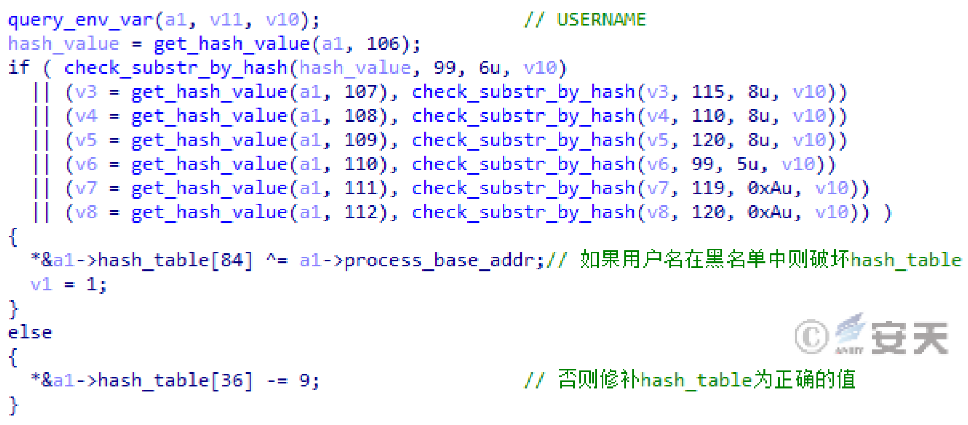
XLoader will detect its own process name and stop running if the process name meets a specific hash value or its length is greater than 31 and does not contain spaces.



```
ImageBaseAddress = get_ImageBaseAddress();
LDR_DATA_TABLE_ENTRY = get_LDR_DATA_TABLE_ENTRY(0, ImageBaseAddress);
wchar_to_char(v10, LDR_DATA_TABLE_ENTRY->BaseDllName.Buffer);
v4 = strlen(v10);
if ( v4 > 31 )
{
  v5 = 0;
  while ( v10[v5] != ' ' )
  {
    if ( ++v5 >= v4 )
    {
      *&a1->hash_table[72] ^= a1->new_1_ntdll_base_addr;// 破坏hash_table
      v1 = 1;
      break;
    }
  }
}
if ( v4 >= 8 )
{
  v8 = &v9[v4];
  hash_value = get_hash_value(a1, 187);       // 0x7C81C71D
  if ( check_hash(hash_value, v8) )
  {
    ++v1;
    *&a1->hash_table[72] ^= a1->new_1_ntdll_base_addr;// 破坏hash_table
  }
}
```

Figure 3-14XLoader detection process name

XLoader will determine whether it is running in a sandbox by detecting the username.



```
query_env_var(a1, v11, v10);                    // USERNAME
hash_value = get_hash_value(a1, 106);
if ( check_substr_by_hash(hash_value, 99, 6u, v10)
  || (v3 = get_hash_value(a1, 107), check_substr_by_hash(v3, 115, 8u, v10))
  || (v4 = get_hash_value(a1, 108), check_substr_by_hash(v4, 110, 8u, v10))
  || (v5 = get_hash_value(a1, 109), check_substr_by_hash(v5, 120, 8u, v10))
  || (v6 = get_hash_value(a1, 110), check_substr_by_hash(v6, 99, 5u, v10))
  || (v7 = get_hash_value(a1, 111), check_substr_by_hash(v7, 119, 0xAu, v10))
  || (v8 = get_hash_value(a1, 112), check_substr_by_hash(v8, 120, 0xAu, v10)) )
{
  *&a1->hash_table[84] ^= a1->process_base_addr;// 如果用户名在黑名单中则破坏hash_table
  v1 = 1;
}
else
{
  *&a1->hash_table[36] -= 9;                    // 否则修补hash_table为正确的值
}
```

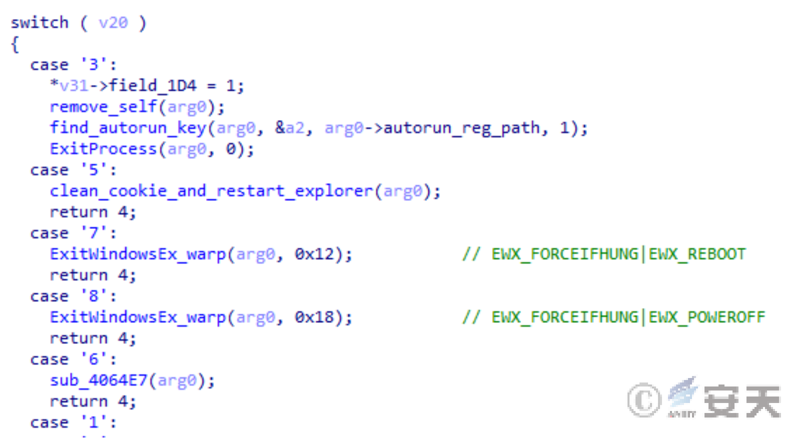
Figure 3-15XLoader checks whether the username is in the blacklist

The specific environment detection list of XLoader is as follows.

Table 3XLoader environment 1XLoader in manually loaded ntdll2Table 5 - 2 List of programs stolen by XLoader

| Outlook | Internet Explorer | F irefox | Thunderbird | Chrome |
|---|---|---|---|---|
| Op era | Foxmail | AVG Secure Browser | Kinza | URBrowser |
| Avast Secure Browser | SalamWeb | CCleaner Browser | Opera Software | YandexBrowser |
| Slimjet | 360Chrome | Comodo Dragon | ChromePlus | Chromium |
| torch | Brave-Browser | Iridium | Opera Neon | 7Star |
| Amigo | Blisk | CentBrowser | Chedot | CocCoc Browser |
| Elements Browser | Epic Privacy Browser | Kometa | Orbitum | Sputnik |
| uCozMedia Uran | Sleipnir5 | Citrio | Coown | liebao |
| QIP Surf | Microsoft Edge | Vivaldi | | |

In the second stage of XLoader, it can also accept and execute C2 instructions.

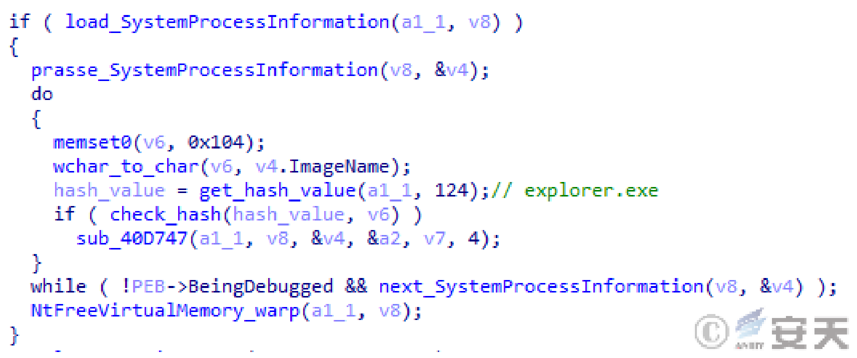


```
switch ( v20 )
{
  case '3':
    *v31->field_1D4 = 1;
    remove_self(arg0);
    find_autorun_key(arg0, &a2, arg0->autorun_reg_path, 1);
    ExitProcess(arg0, 0);
  case '5':
    clean_cookie_and_restart_explorer(arg0);
    return 4;
  case '7':
    ExitWindowsEx_warp(arg0, 0x12);           // EWX_FORCEIFHUNG|EWX_REBOOT
    return 4;
  case '8':
    ExitWindowsEx_warp(arg0, 0x18);           // EWX_FORCEIFHUNG|EWX_POWEROFF
    return 4;
  case '6':
    sub_4064E7(arg0);
    return 4;
  case '1':
```

Figure 3-16XLoader executes C2 instructions

| Instruction number | Function |
|---|---|
| 1 | Save and execute the executable file sent by C2 through ShellExecuteA |
| 2 | XLoader in the system |
| 3 | XLoader in the system |
| 4 | Execute the specified program through ShellExecuteA |
| 5 | Clean Cookies files and restart explorer.exe |
| 6 | Re-execute the stealing logic |
| 7 | Restart the target computer |
| 8 | Shut down the target computer |

XLoader will then inject itself into exploiter.exe to run the third stage code.

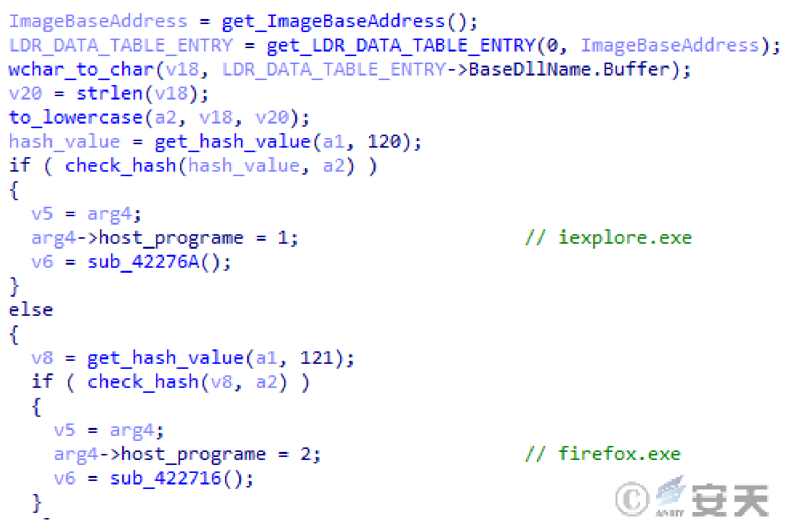


```
if ( load_SystemProcessInformation(a1_1, v8) )
{
  prasse_SystemProcessInformation(v8, &v4);
  do
  {
    memset0(v6, 0x104);
    wchar_to_char(v6, v4.ImageName);
    hash_value = get_hash_value(a1_1, 124);// explorer.exe
    if ( check_hash(hash_value, v6) )
      sub_40D747(a1_1, v8, &v4, &a2, v7, 4);
  }
  while ( !PEB->BeingDebugged && next_SystemProcessInformation(v8, &v4) );
  NtFreeVirtualMemory_warp(a1_1, v8);
}
```

Figure 3-17XLoader is injected into explorere.exe to execute the third stage logic

## 3.3 XLoader Third Stage

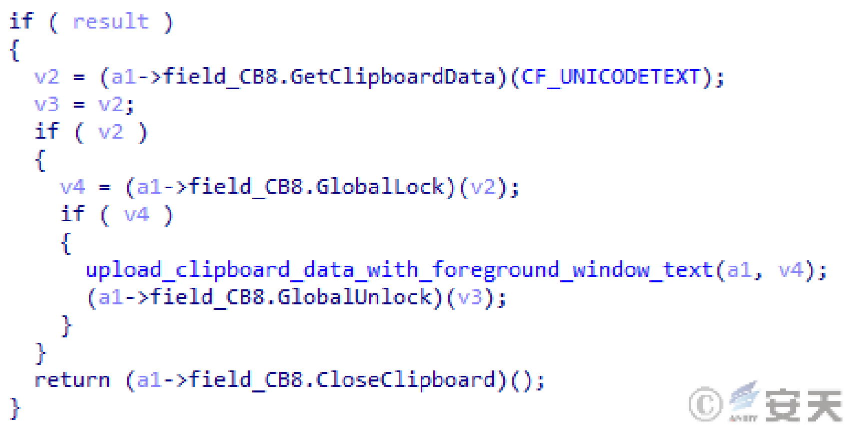
In the third stage of initialization, XLoader does not check the running environment as in the first two stages, but only loads the necessary information for running. After the initialization of XLoader, it will detect the currently injected program.



```
ImageBaseAddress = get_ImageBaseAddress();
LDR_DATA_TABLE_ENTRY = get_LDR_DATA_TABLE_ENTRY(0, ImageBaseAddress);
wchar_to_char(v18, LDR_DATA_TABLE_ENTRY->BaseDllName.Buffer);
v20 = strlen(v18);
to_lowercase(a2, v18, v20);
hash_value = get_hash_value(a1, 120);
if ( check_hash(hash_value, a2) )
{
  v5 = arg4;
  arg4->host_programe = 1;                    // iexplore.exe
  v6 = sub_42276A();
}
else
{
  v8 = get_hash_value(a1, 121);
  if ( check_hash(v8, a2) )
  {
    v5 = arg4;
    arg4->host_programe = 2;                  // firefox.exe
    v6 = sub_422716();
  }
```

Figure 3-18XLoader detects the currently injected program

In XLoader analyzed this time, XLoader will only perform malicious behavior when injected into ex plorer.exe . At this time, XLoader will create a thread to steal clipboard and window information.



```
if ( result )
{
  v2 = (a1->field_CB8.GetClipboardData)(CF_UNICODETEXT);
  v3 = v2;
  if ( v2 )
  {
    v4 = (a1->field_CB8.GlobalLock)(v2);
    if ( v4 )
    {
      upload_clipboard_data_with_foreground_window_text(a1, v4);
      (a1->field_CB8.GlobalUnlock)(v3);
    }
  }
  return (a1->field_CB8.CloseClipboard)();
}
```

Figure 3-19XLoader steals clipboard information

## 4 IoCs

| IoCs |
|---|
| 3CD23779C02B9A35DE36B1DAFA13C268 |
| 8D80D388BE65DFAF7BBD1CA61D87CADE |
| 14321BB37BA45223C9BE4633CA4E11B9 |
| 59A95C9FCCA51D487259BA447380AD41 |

## 5 ATT&CK Mapping of Samples

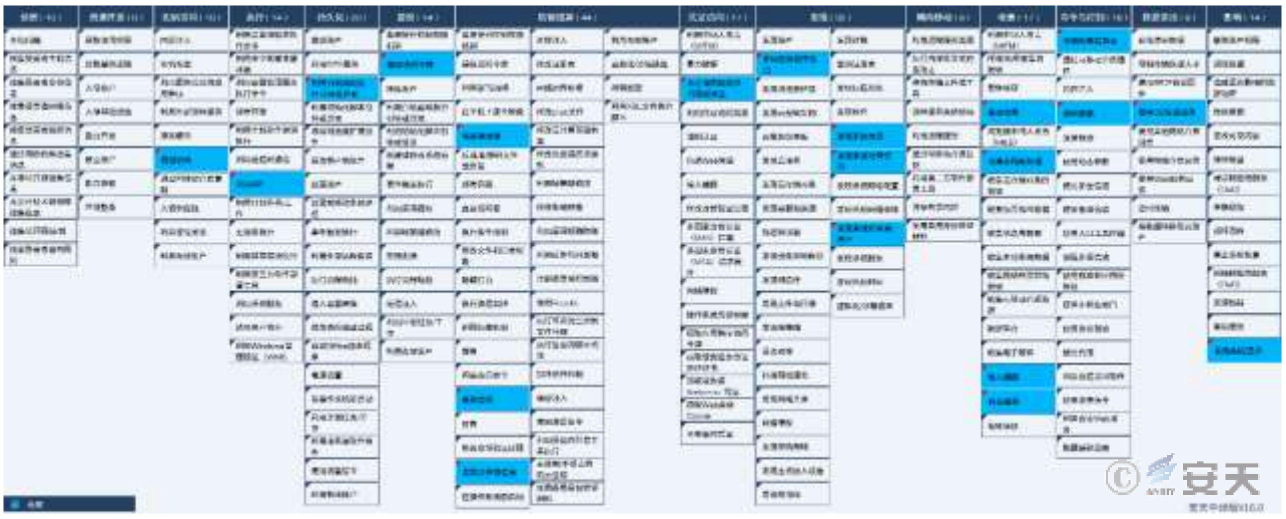


Figure 5-1technical features to ATT&CK

Specific ATT&CK technical behavior description table:

Table 7-1 ATT&CK technical behavior description table

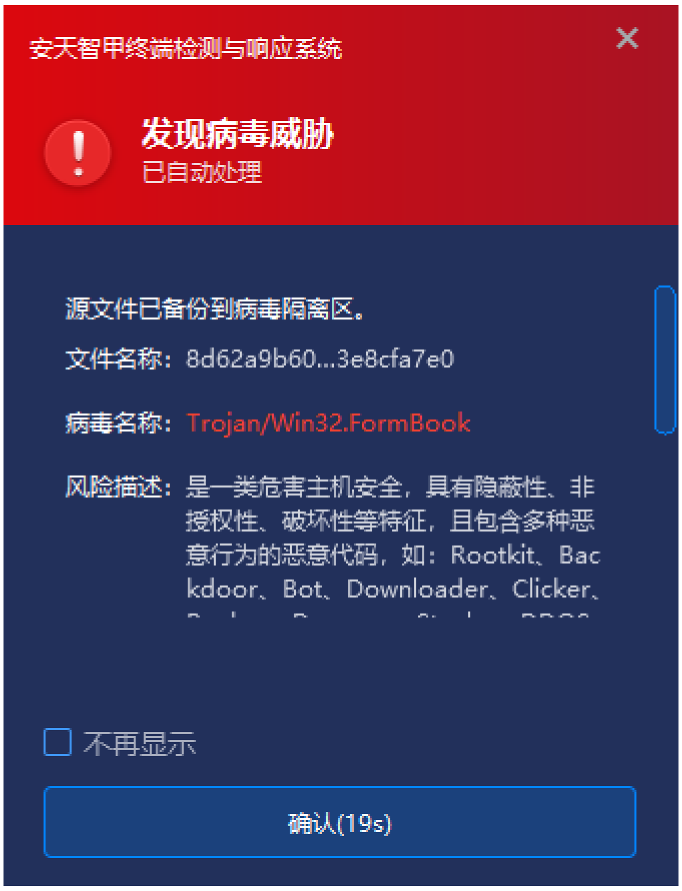
| ATT&CK Stages/Categories | Specific Behavior | Notes |
|---|---|---|
| **Initial visit** | Phishing | XLoader achieves initial access through phishing emails |
| **Execute** | Leverage APIs | XLoader relies on process hollowing and APC injection to execute different stages of payloads |
| **Persistence** | Boot or log in with autostart | XLoader achieves persistence by modifying the registry |
| **Privilege escalation** | Manipulate access tokens | XLoader obtains SeDebugPrivilege permissions through the AdjustTokenPrivilege function |
| **Defense evasion** | Avoid debuggers | XLoader detects whether WOW32Reserved is hooked |
| | | XLoader detects whether there is a kernel-level debugger and debug port through NtQuerySystemInformation |

| | | |
|---|---|---|
| | | XLoader detects whether there is a process/module/user name matching the blacklist |
| | Delete a beacon | XLoader will erase the PE header information of the injected program to avoid detection |
| | | XLoader will delete the files used for initial access |
| | | XLoader can be uninstalled in remote control function |
| | Obfuscate files or information | Most of the code and data of XLoader require multiple differentials and RC 4 for decryption |
| | | XLoader dynamically loads API by calculating function hash |
| | | XLoader important data needs to be read in a complex way to discard useless bytes |
| | | XLoader will re-encrypt the function after it is used. |
| | | XLoader decryption code and the key of the data block are XOR-encrypted |
| **Credential access** | Get the credentials from where the password is stored | Passwords from various browsers, email clients, and Windows Credentials Vault |
| **Discover** | Discovery Application Window | XLoader will record the window information of the hooked program |
| | Discover system information | XLoader will collect system version information |
| | Discover the system's geographic location | XLoader will decide whether to run based on the system's geographical location |
| | Discover the system owner/user | XLoader will obtain the username and upload it to the C2 server |
| **Collect** | Automatic Collection | TCP and HTTP traffic sent by the program through hooks. |
| | Collect Clipboard Data | XLoader collects clipboard data through the GetClipboardData function |
| | Input Capture | XLoader will hook the window message function to capture keyboard input |
| | Screen Capture | XLoader will take a screenshot and send it back to the C2 server |
| **Command and control** | Use application layer protocols | XLoader uses HTTP protocol to communicate with the C2 server |
| | Encode Data | XLoader will encrypt the message with RC4+base64 |
| **Data exfiltration** | Use C2 channel for backhaul | XLoader uses the same channel as C2 to transmit stolen information |
| **Influence** | System shutdown/restart | XLoader can shut down/restart the computer |

## 6 Recommendations for Protection

It is recommended that enterprise users deploy professional terminal security protection products, conduct real-time detection of local new and startup files, and periodically perform virus scans within the network. Antiy Intelligent Endpoint Protection System series products (hereinafter referred to as "IEP") rely on Antiy's self-developed threat detection engine and kernel-level active defense capabilities to effectively detect and kill the virus samples discovered this time.

IEP can monitor local disks in real time, automatically detect viruses on newly added files, send alerts and handle viruses as soon as they are discovered, and prevent malicious code from being activated.


安天智甲终端检测与响应系统

发现病毒威胁
已自动处理

源文件已备份到病毒隔离区。
文件名称：8d62a9b60...3e8cfa7e0
病毒名称：Trojan/Win32.FormBook
风险描述：是一类危害主机安全，具有隐蔽性、非授权性、破坏性等特征，且包含多种恶意行为的恶意代码，如：Rootkit、Backdoor、Bot、Downloader、Clicker、

Figure 6-1When a virus is found, IEP captures it and sends an alert immediately

IEP also provides users with a unified management platform, through which administrators can centrally view the details of threat events within the network and handle them in batches, thereby improving the efficiency of terminal security operation and maintenance.

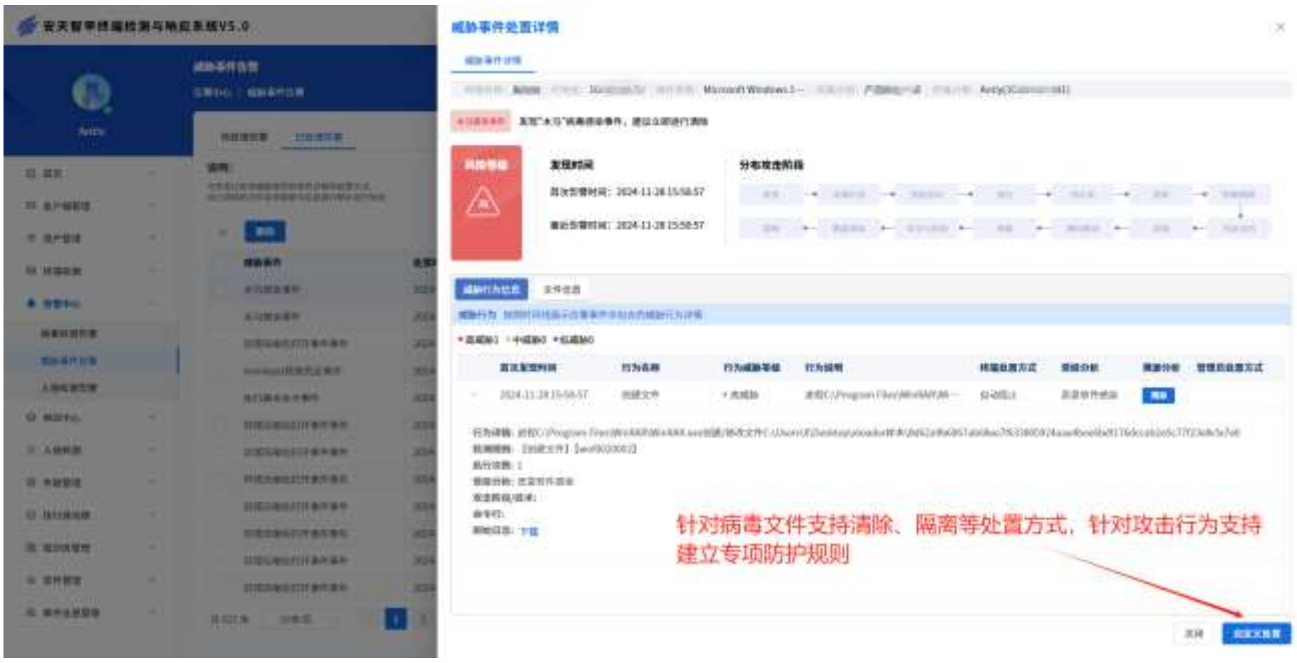

针对病毒文件支持清除、隔离等处置方式，针对攻击行为支持建立专项防护规则

Figure 6-2View and complete threat event handling through the IEP Management Center

## 7 Antiy LanDi VILLM

Antiy LanDi VILLM can automatically analyze samples. Some of the analysis contents are as follows:

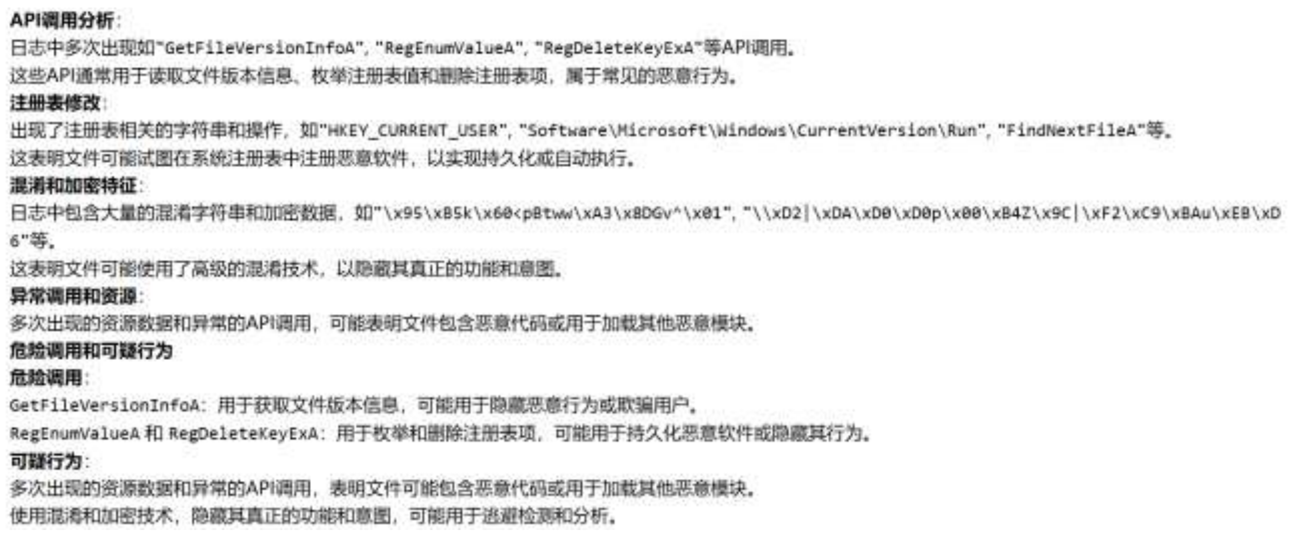

**API调用分析**:
日志中多次出现如"GetFileVersionInfoA", "RegEnumValueA", "RegDeleteKeyExA"等API调用。
这些API通常用于读取文件版本信息、枚举注册表值和删除注册表项，属于常见的恶意行为。
**注册表修改**:
出现了注册表相关的字符串和操作，如"HKEY_CURRENT_USER", "Software\Microsoft\Windows\CurrentVersion\Run", "FindNextFileA"等。
这表明文件可能试图在系统注册表中注册恶意软件，以实现持久化或自动执行。
**混淆和加密特征**:
日志中包含大量的混淆字符串和加密数据，如"\x95\xB5k\x60<pBtww\xA3\x8DGv^\x01", "\\xD2|\xDA\xD0\xD0p\x00\xB4Z\x9C|\xF2\xC9\xBAu\xEB\xD6"等。
这表明文件可能使用了高级的混淆技术，以隐藏其真正的功能和意图。
**异常调用和资源**:
多次出现的资源数据和异常的API调用，可能表明文件包含恶意代码或用于加载其他恶意模块。
**危险调用和可疑行为**
**危险调用**:
GetFileVersionInfoA: 用于获取文件版本信息，可能用于隐藏恶意行为或欺骗用户。
RegEnumValueA 和 RegDeleteKeyExA: 用于枚举和删除注册表项，可能用于持久化恶意软件或隐藏其行为。
**可疑行为**:
多次出现的资源数据和异常的API调用，表明文件可能包含恶意代码或用于加载其他恶意模块。
使用混淆和加密技术，隐藏其真正的功能和意图，可能用于逃避检测和分析。

Figure 7-Antiy LanDi VILLM sample analysis interpretation

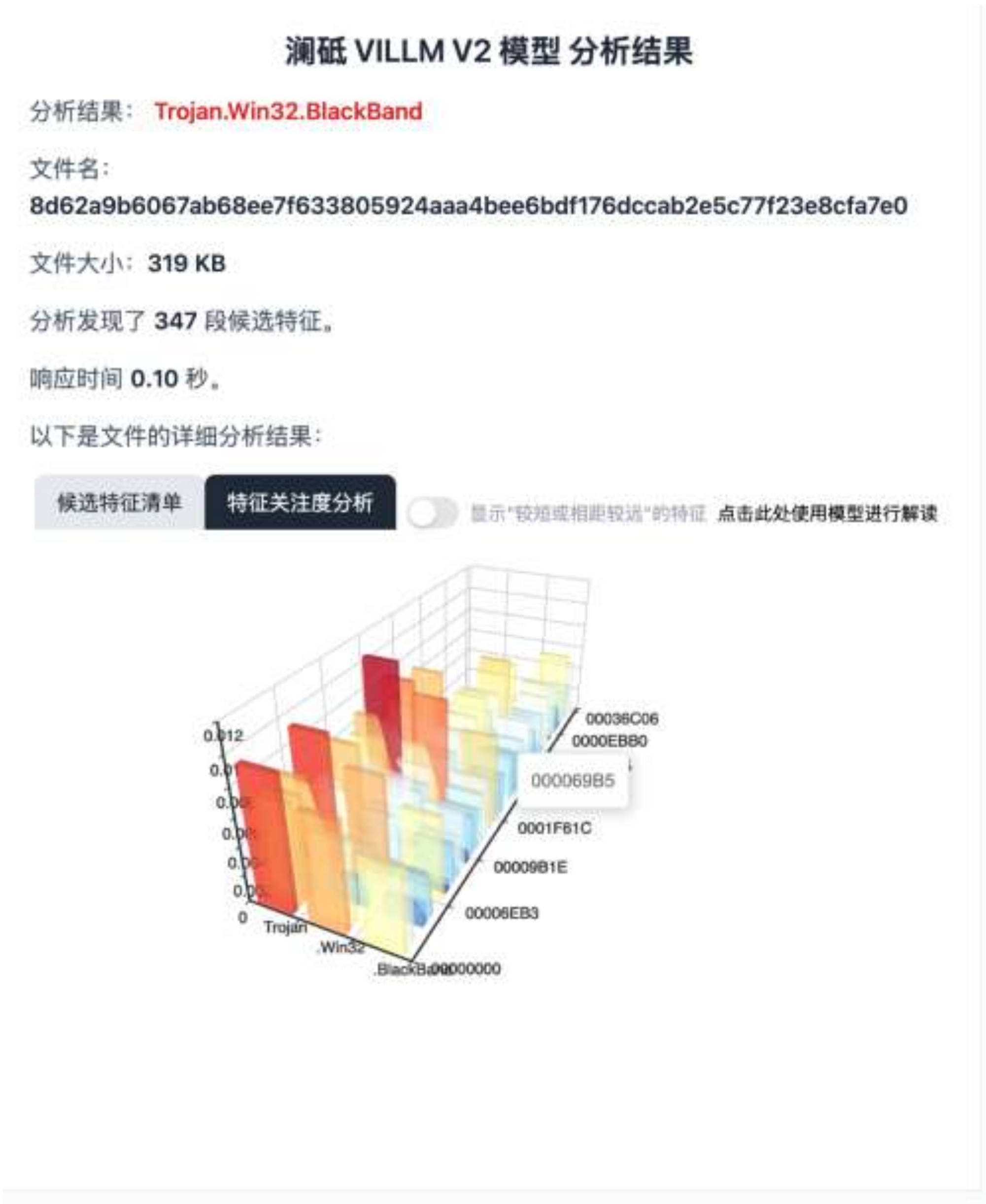
According to the draft national standard, the threat classification is Trojan.Win32.BlackBand, and the YARA rules generated by automatic feature extraction are as follows:

rule Trojan_Win32_BlackBand:Trojan

{ meta:

description = " This is a YARA rule generated by the VILLM V2 model. It is used to detect Trojan.Win32.BlackBand"; strings: $a = {53 51 53 53 53 50 } condition: $a}

Antiy LanDi VILLM is the first threat detection generative algorithm in China that has been registered with the Cyberspace Administration of China. The model is trained based on the massive sample feature engineering data accumulated by Antiy Cyber Super Brain for more than 20 years. The training data includes file identification information, judgment information, attribute information, structure information, behavior information, host environment information, data information, etc. It supports threat judgment and output of detailed knowledge understanding of vector features in different scenarios, forming a multi-modal detection method for different application needs and scenarios, improving the background hidden threat judgment ability, and further empowering security operations.



澜砥 VILLM V2 模型 分析结果

分析结果： Trojan.Win32.BlackBand

文件名：
8d62a9b6067ab68ee7f633805924aaa4bee6bdf176dccab2e5c77f23e8cfa7e0

文件大小： 319 KB

分析发现了 347 段候选特征。

响应时间 0.10 秒。

以下是文件的详细分析结果：

Figure 7-1Sample analysis results of Antiy LanDi VILLM

## Appendix 1: References

[1]. Antiy. Analysis Report on the Attack of Formbook Spyware Malware on a Certain Organization[R/OL].(2021-10-21)

https://www.antiy.cn/research/notice&report/research_report/20211021.html

[2]. Antiy.Trojan/Win32.XLoader Virus Detailed Explanation and Protection - Computer Virus Encyclopedia[R/OL].(2025-01-22)

https://www.virusview.net/malware/Trojan/Win32/XLoader

## Appendix 2: About Antiy

Antiy is committed to enhancing the network security defense capabilities of its customers and effectively responding to security threats. Through more than 20 years of independent research and development, Antiy has developed technological leadership in areas such as threat detection engines, advanced threat countermeasures, and large-scale threat automation analysis.

Antiy has developed IEP (Intelligent Endpoint Protection System) security product family for PC, server and other system environments, as well as UWP (Unified Workload Protect) security products for cloud hosts, container and other system environments, providing system security capabilities including endpoint antivirus, endpoint protection (EPP), endpoint detection and response (EDR), and Cloud Workload Protection Platform (CWPP) , etc. Antiy has established a closed-loop product system of threat countermeasures based on its threat intelligence and threat detection capabilities, achieving perception, retardation, blocking and presentation of the advanced threats through products such as the Persistent Threat Detection System (PTD), Persistent Threat Analysis System (PTA), Attack Capture System (ACS), and TDS. For web and business security scenarios, Antiy has launched the PTF Next-generation Web Application and API Protection System (WAAP) and SCS Code Security Detection System to help customers shift their security capabilities to the left in the DevOps process. At the same time, it has developed four major kinds of security service: network attack and defense logic deduction, in-depth threat hunting, security threat inspection, and regular security operations. Through the Threat Confrontation Operation Platform (XDR), multiple security products and services are integrated to effectively support the upgrade of comprehensive threat confrontation capabilities.

Antiy provides comprehensive security solutions for clients with high security requirements, including network and information authorities, military forces, ministries, confidential industries, and critical information infrastructure. Antiy has participated in the security work of major national political and social events since 2005 and has won honors such as the Outstanding Contribution Award and Advanced Security Group. Since 2015, Antiy's products and services have provided security support for major spaceflight missions including manned spaceflight, lunar exploration, and space station docking, as well as significant missions such as the maiden flight of large aircraft, escort of main force ships, and Antarctic scientific research. We have received several thank-you letters from relevant departments.

Antiy is a core enabler of the global fundamental security supply chain. Nearly a hundred of the world's leading security and IT enterprises have chosen Antiy as their partner of detection capability. At present, Antiy's threat detection engine provides security detection capabilities for over 1.3 million network devices and over 3 billion smart terminal devices worldwide, which has become a "national-level" engine. As of now, Antiy has filed 1,877 patents in the field of cybersecurity and obtained 936 patents. It has been awarded the title of National Intellectual Property Advantage Enterprise and the 17th (2015) China Patent Excellence Award.

Antiy is an important enterprise node in China emergency response system and has provided early warning and comprehensive emergency response in major security threats and virus outbreaks such as "Code Red", "Dvldr", "Heartbleed", "Bash Shellcode" and "WannaCry". Antiy conducts continuous monitoring and in-depth analysis against dozens of advanced cyberspce threat actors (APT groups) such as "Equation", "White Elephant", "Lotus" and "Greenspot" and their attack actions, assisting customers to form effective protection when the enemy situation is accurately predicted.